

**Appendix H:
WAVE/DSRC Security**

TABLE OF CONTENTS

1	Background	1
2	Threat Model.....	2
2.1	Background.....	2
2.2	Introduction.....	2
2.3	Overview of VSC for the Purpose of a Security Study	3
2.3.1	Curve Speed Warning	4
2.3.2	Traffic Signal Violation Warning.....	5
2.3.3	Extended Brake Lights	6
2.3.4	Other Potential Situations.....	6
2.3.5	VSC Unit Life Cycle	6
2.4	Desired Security Services	7
2.4.1	Message Integrity / Origin Authentication (MI /OA).....	7
2.4.2	Correctness	7
2.4.3	Privacy.....	7
2.4.4	Robustness.....	8
2.4.5	Fail-Safety	8
2.5	A Strawman Security Architecture for Use in the Threat Model	9
2.5.1	Providing Communication Security	9
2.5.2	Correctness	10
2.6	Required Unit Capabilities.....	11
2.6.1	OBUs.....	11
2.6.2	RSUs.....	11
2.7	Attacker Capabilities.....	12
2.7.1	Class 1: Attackers With a Programmable Radio Transmitter/Receiver	13
2.7.2	Class 2: Attackers With Access to an Unmodified VSC Unit.....	14
2.7.3	Class 3: Attackers Who Have Recovered Keying Material From a VSC Unit	16
2.7.4	Class 4: 'Inside' Attackers.....	17
2.7.5	Some Out-of-Scope Threats	19
2.8	Containing Compromise	19
2.8.1	Proactive Containment	20
2.8.2	Discriminators for Compromised Units	20
2.8.3	Residual Capabilities of Compromised Units	20

3	Constraints	21
3.1	Introduction.....	21
3.2	Network Characteristics.....	21
3.2.1	Total Number of Units	22
3.2.2	Maximum Data Rate	22
3.2.3	Packets Per Second.....	22
3.2.4	Latency.....	22
3.2.5	Packet Loss Rate	23
3.2.6	Packet Size	23
3.3	Environmental Characteristics	23
3.3.1	Unit Size.....	24
3.3.2	Temperature Range	24
3.3.3	Heat Output	24
3.3.4	Power Consumption	24
3.4	Cost of Goods	25
3.4.1	RSUs.....	25
3.4.2	OBUs.....	25
3.4.3	Management Consoles	25
3.4.4	Service Consoles	25
3.4.5	Manufacturing Facilities.....	26
3.5	Management Costs.....	26
3.5.1	Investigation of Compromise	26
3.5.2	Key Infrastructure.....	26
3.5.3	Management of RSUs	28
4	Architecture	29
4.1	Introduction.....	29
4.2	Architecture Summary	29
4.3	Architectural Overview.....	29
4.3.1	Key Hierarchy	29
4.3.2	Communications Architecture.....	30
4.3.3	Message Authentication	31
4.3.4	Management, Revocation, and Updates	32
4.3.5	Fail-Open versus Fail-Closed.....	32

4.4 RSUs and Public Safety OBUs	32
4.4.1 Authentication Structure.....	33
4.4.2 Certificate Format	35
4.4.3 Registration and Certificate Issuance	37
4.4.4 Message Authentication	38
4.4.5 Propagation of Certificates	40
4.4.6 Identifying Compromised Units.....	40
4.4.7 Certificate Revocation.....	41
4.4.8 Choice of Algorithms	41
4.5 Authentication for OBU Messages	44
4.5.1 Selection Criteria.....	44
4.5.2 Building Blocks.....	46
4.5.3 Global Symmetric Keys	50
4.5.4 Public Key Signature Based Schemes.....	51
4.5.5 Group Signatures.....	62
4.5.6 Comparison of Schemes.....	62
5 Protocol.....	64
5.1 Introduction.....	64
5.2 Protocol Summary	64
5.3 Protocol Definition.....	64
5.3.1 Certificate Format	64
5.3.2 Certificate Revocation List Format	73
5.3.3 VSP Message Format	75
5.3.4 Signed Message Format	76
5.3.5 System Updates	82
5.4 Enrollment Procedures.....	84
5.4.1 CAs, RSUs, and PSOBUs	84
5.4.2 OBUs Certificate Initialization.....	86
5.5 Suggested Operating Procedures	86
5.5.1 OBU Operating Requirements	86
5.5.2 OBU CAs and Escrow Authorities.....	87
5.5.3 CA's Procedures.....	87
5.6 OBU Privacy Parameters	87
5.6.1 Choice of Work Factor for OBU Blinding.....	87
5.6.2 OBU Certificates	88

5.7 Opportunities for Optimization.....	88
5.7.1 Combined Algorithm Identifiers	88
5.7.2 Length of Fields	89
5.8 Summary of Syntax.....	89
6 References.....	97
7 Exhibit A: Analysis of ECDSA Performance.....	100
8 Exhibit B: Choice of Key Size	102
9 Exhibit C: The Subset Difference Algorithm.....	103
10 Exhibit D: Table of Acronyms	105

LIST OF TABLES

Table 1: Summary of Attacker Types and Attacks.....	19
Table 2: Summary of Network Constraints.....	23
Table 3: Summary of Environmental Constraints.....	25
Table 4: Message Sizes for Blind Signatures	57
Table 5: ECDSA Key Strengths and Sizes.....	102

LIST OF FIGURES

Figure 1: Curve Speed Warning Scenario	4
Figure 2: Traffic Signal Violation Warning Scenario	5
Figure 3: Extended Brake Lights Scenario.....	6
Figure 4: Strawman Key Distribution	9
Figure 5: Strawman Block Diagram.....	10
Figure 6: Signal/Message Conflict	15
Figure 7: Overview of Dual Authentication Structure	30
Figure 8: VSC Communications Architecture.....	31
Figure 9: A Protected Message.....	31
Figure 10: Functional Scope for RSUs and Public Safety OBUs.....	37

Figure 11: A Protected Message.....	38
Figure 12: A Broadcast Encryption Tree.....	46
Figure 13: Erasure Code Transmission and Reconstruction	48
Figure 14: A Trivial Code	49
Figure 15: Innocent Revocation Rates	60
Figure 16: Subset Difference Broadcast Tree.....	76
Figure 17: Subset Difference Broadcast Tree.....	103
Figure 18: Keys Assigned to Node $N_{l,r}$	104

1 Background

Security is an important consideration for Dedicated Short Range Communications (DSRC) that supports vehicle-safety applications. For the system to be secure, the applications must be able to trust that the communication has been received unaltered and from a known source. The communication should require a low amount of computational and communications overhead, and it must be robust in the event of individual units being compromised. Most of the vehicle-safety applications studied in the VSC project broadcast messages to all receivers, rather than directed to a given peer, which creates additional security challenges.

Communications security consultants completed the majority of technical work for this task, as well as the majority of the documentation contained in this appendix. The solutions presented in this appendix represent the recommendations of the consultants, which appear to meet the assumptions and constraints within this task. However, with the currently proposed architecture, which includes many optimizations, each message transmitted would include significant overhead, and the message signatures would take time to process once they are received. Management of a Public Key Infrastructure for RSUs would also be necessary, according to the proposed scheme. This analysis represents a foundational examination of the security question, and provides a recommendation upon which future testing may be accomplished. The overall feasibility of implementing the proposed security approach has not yet been determined.

2 Threat Model

2.1 Background

Security is a primary concern for vehicle-safety applications that are enabled or enhanced by wireless communications. This is the first in a series of sections describing the security issues associated with Vehicle Safety Communications and their possible solutions. This section focuses on the threat model, which will need to be considered by any potential design.

In this threat model, we identified four major security goals for the VSC system:

1. Message integrity/origin authentication – receivers can tell that a valid unit generated the message(s);
2. Correctness – when units generate valid messages, we can have some assurance that they were correct;
3. Privacy – it should not be possible to use VSC to remotely obtain private information about a vehicle's behavior and then tie it back to a single identified vehicle; and
4. Robustness – the system should contain mechanisms for containing misbehaving units and, where possible, continue to operate under attack.

In order to accomplish these security goals, we need to consider the kinds of attacks that might be mounted. We contemplated four basic kinds of attackers with escalating capabilities:

1. Attackers with a programmable radio transmitter/receiver;
2. Attackers with access to an un-modified VSC unit who can, therefore, control the inputs, sensors, etc.;
3. Attackers who have access to a modified VSC unit and who have obtained the keying material; and
4. "Inside" attackers who have access to records and equipment operated by the vehicle manufacturer or the VSC unit manufacturer.

Each of these attackers has a certain class of attacks that they are able to mount. In general, attackers with greater capabilities will also be able to mount a greater number of attacks, including potentially more damaging ones. However, certain kinds of attacks – such as those on the Global Positioning System (GPS) satellites and ground stations – are out of scope for this threat model. Attackers with the capability to mount such attacks can, in general, not be defended against, though there are techniques available to contain the threat they present.

We must assume throughout our design that some attacks will succeed and some units will be compromised. Thus, the design of the eventual system must incorporate this assumption as part of the threat model. One major focus of future work will be on containing compromise by detecting when it has occurred and shutting compromised units out of the VSC system.

2.2 Introduction

5.9 GHz DSRC is a wireless communications system that provides vehicle-vehicle and infrastructure-vehicle communications services. One of the anticipated major uses of this technology is for safety information such as emergency braking warnings, traffic signal violation warnings, and curve speed warnings.

Security is a primary concern in any vehicle-safety application and it is especially important when the critical data is being transmitted over the air. Without security, an attacker could tamper with

the messages being communicated and potentially cause harm. To take merely one example, an attacker who could forge messages could send false braking indicators, causing inappropriate emergency braking and perhaps crashes.

The first step in designing any security system is to determine the threat model. A threat model describes the capabilities that an attacker is assumed to be able to deploy against the system. It then attempts to predict what sorts of attacks those capabilities might enable the attacker to mount. Equally important, it specifies the forms of attack that are out of scope for the system. Nearly every security system is vulnerable to a sufficiently dedicated and resourceful attacker. The goal of security is not to provide absolute protection but to manage risk. When deciding whether to protect against an attack, the difficulty of providing that protection needs to be balanced against the seriousness of the threat and the difficulty of mounting it.

It is extremely difficult to estimate the magnitude of each threat, since the potential harm can vary widely. For example, if an attacker could impersonate an emergency vehicle, he or she could mount threats with consequences of different magnitudes. The attacker could use the communication to move quickly through traffic, convincing drivers that an emergency vehicle is approaching and causing them to move to the side of the road. This could result in delays and possibly confusion (when they do not see an emergency vehicle) for the other drivers. That same attacker could alternatively plan to preempt traffic signals in a coordinated fashion to make sure that the vehicle of an elected official, for example, would be properly placed for a terrorist attack. This scenario clearly has more serious consequences. Taking another example, if an attacker can impersonate a vehicle in an emergency braking maneuver, he or she could cause traffic jams behind them that could serve several purposes. The disturbance could simply be an amusement for the attacker, or it could be used in collaboration with other attackers to temporarily obstruct traffic arteries around a targeted city. Because the potential consequences of an attack vary so widely, they are not quantified in this report. However, some recent real-world examples may serve to highlight the need for secure communications.

In July of 2003, Kohno et al. [41] published a series of flaws in Diebold Election Systems's electronic voting system, causing some loss of confidence in e-voting and significant negative publicity for Diebold, which has remained in the news long after the initial publication. Secondly, 3M currently sells a device which allows traffic signals to be remotely controlled by public-safety vehicles. This system has no security, and consumer units that can control these intersections recently have become available [42], much to the dismay of the press and public. If VSC does not include the appropriate security, similar results are possible.

2.3 Overview of VSC for the Purpose of a Security Study

In this section, VSC is described briefly in terms pertinent to a security study. Reporting for the threat model will refer to the VSC system, as described in this section.

The VSC system will allow vehicles to communicate with each other and with infrastructure elements. Broadly speaking, this means there are two types of network elements:

Road Side Units (RSUs) – network nodes embedded in infrastructure elements such as road signs, traffic signals, etc., and

On-Board Units (OBUs) – network nodes embedded in vehicles.

In both cases, it is expected that the units will at a minimum consist of:

- A general purpose processor and associated memory;
- A radio transmitter and receiver;

- Interfaces to sensors as required; and
- A GPS receiver (for non-stationary units).

This combination of elements enables a number of safety application scenarios [1]. For purposes of analysis we will focus on three such application scenarios, which were determined to be representative of the required system capabilities. We briefly describe those scenarios here:

2.3.1 Curve Speed Warning

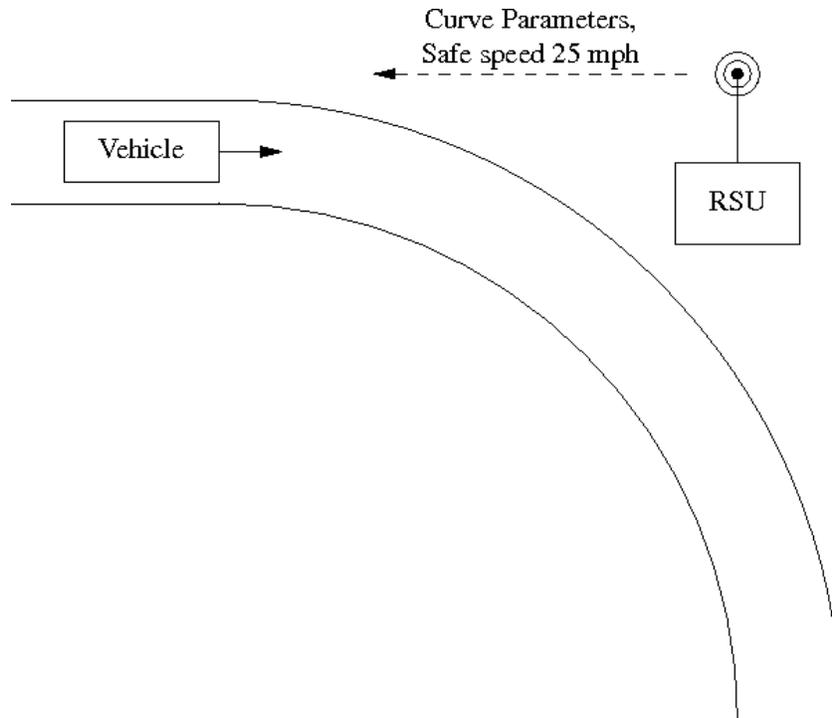


Figure 1: Curve Speed Warning Scenario

Probably the simplest VSC application scenario is *Curve Speed Warning* (CSW). It is quite common for curves in roads to have safe traversal speeds that are significantly less than the current speed limit or, at least, less than the prevailing vehicle speed. Such curves often have signs posted that warn drivers of the maximum safe speed for that curve. CSW is effectively an electronic version of such signs. Since radio transmissions can be received beyond line of sight, CSW can provide advance warning of the need to slow down.

In its simplest form, CSW could be implemented as a radio transmitter broadcasting the same speed limit as posted on the sign. As currently contemplated, it would also include the shape of the oncoming curve/curves so that the vehicle can calculate individual limits. The user interface for CSW is not specified; however for our purposes we assume that when the OBU receives a CSW signal it attempts to determine whether the vehicle is moving at a safe speed, and if not, warns the driver in some way.

2.3.2 Traffic Signal Violation Warning

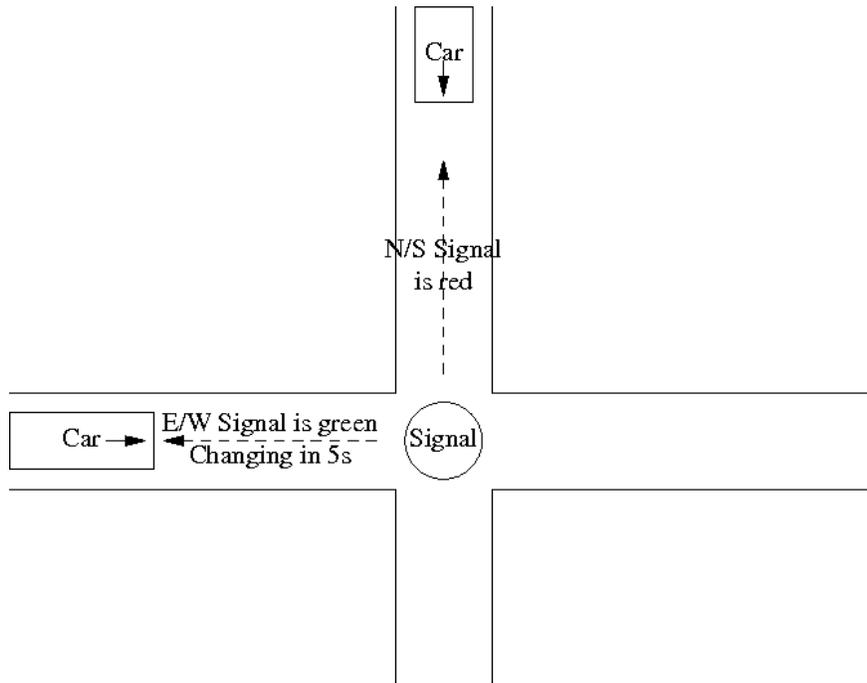


Figure 2: Traffic Signal Violation Warning Scenario

Traffic Signal Violation Warning (TSVW) acts to increase the amount of information provided by traffic signals and allow the driver to be warned if he is in danger of going through a red light. Traditional traffic signals have three modes: red, green, and yellow. The current color state provides only a limited amount of information about the time until the next state. In particular, the time from red to green and green to red is almost entirely unbounded. The TSVW application scenario provides this timing information to the vehicle, which can warn the driver when appropriate.

A TSVW RSU would likely be attached to a traffic signal. It would periodically broadcast its own location (to identify the specific traffic signal being referenced), the current state of the signal in each direction, plus the expected time until the next state and what that state will be. As with the CSW application scenario, the OBU would attempt to determine whether the vehicle was in danger of violating the traffic signal and notify the driver if so.

2.3.3 Extended Brake Lights

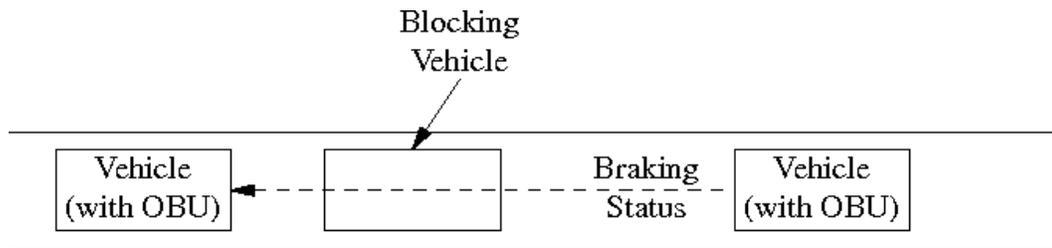


Figure 3: Extended Brake Lights Scenario

Like CSW and TSVW, *Extended Brake Lights* (EBL) extends the operational range of conventional visual signals. Unlike CSW and TSVW, EBL is a vehicle-to-vehicle application scenario rather than an infrastructure-vehicle application scenario. The purpose of EBL is to extend the range of brake light signals in emergency braking situations. Currently, drivers can see when the vehicle ahead of them is braking, but cannot see the brake lights of vehicles further down the road. In addition, brake lights are either on or they are off. They provide no information about the magnitude of the deceleration. These factors limit drivers' reaction time to emergency braking events.

The purpose of EBL is to enable improved response to such events. When a VSC-enabled vehicle detects hard braking (via brake pedal sensors, accelerometers, etc.) it broadcasts an EBL warning containing its position, velocity, acceleration, etc. When other VSC-enabled vehicles in the vicinity receive the warning, their onboard CPUs determine whether this is a potential threat and, if so, warn the driver of the situation.

Note that unlike CSW and TSVW, in which the OBUs are receivers, EBL requires the OBU to have sensors and a radio transmitter.

2.3.4 Other Potential Situations

We have mentioned two kinds of messages: vehicle-to-vehicle and infrastructure-to-vehicle. For completeness, we note that there are two other potential kinds of messages: vehicle-to-infrastructure (e.g. for traffic density measurement) or infrastructure-to-infrastructure (for coordination purposes). We do not believe that these situations introduce new security threats beyond those in the situations described above.

2.3.5 VSC Unit Life Cycle

OBUs are likely to have a simple life cycle: they are manufactured – either by the vehicle manufacturer or a subcontractor – and then installed in the vehicle. At some point during this process they are initialized with the appropriate keying material. They then require no real management other than keying material (and potentially software) updates, and deactivation under certain removal, reuse and vehicle retirement conditions. The means and criteria for deactivation are not defined in this report and need to be developed and tested via a detailed FMEA process. One exception here is public safety OBUs, which derive their authority from the agency that operates their vehicles, not the vehicle manufacturer. Their life cycle is much more

like that of RSUs, and the details of their life cycle management will be covered by a different team (see Section 2.1, entitled “Interaction With DSRC Standards Groups”).

The life cycle of RSUs is more complicated than that of ordinary OBUs. The first difference is that the RSU derives its authority from the agency that operates it. Thus, RSUs will likely be initialized by the transportation agency that installs them. The second difference is that RSUs may need to be reconfigured as conditions change; for instance, a stop sign might be moved. Thus, RSUs must be field-manageable or even remotely manageable via the Internet. This management system must be constructed in such a way that only authorized agents can manage the RSU. This sort of authentication, authorization, and access control is a classic security problem with a number of known solutions.

2.4 Desired Security Services

The most basic security service that we are interested in providing is that the receiver of a VSC message obtains an accurate picture of the state of the world, as far as the transmitter knew it. It is easiest to think of this service as rejecting all messages that do not match those which would be generated by a perfect VSC unit. Imagine that each vehicle and infrastructure unit were fitted with a perfect VSC unit that always broadcast correct messages. The goal of VSC security, then, is to arrange that any given VSC receiver can reject any messages that do not match those which would have been generated by the perfect VSC unit.

We should be able to obtain this goal even in the face of attack. Attackers might have a variety of capabilities, which we consider in Section 3.7. Most likely this includes the ability to broadcast arbitrary messages and/or the ability to cause some units to lie. Thus, while we must expect that receiving units will receive bogus messages that do not match those which would be transmitted by the perfect unit, the unit should be able to reject those messages.

This general security service implies two sub-services: message integrity/origin authentication and correctness.

2.4.1 Message Integrity / Origin Authentication (MI/OA)

The purpose of MI/OA is to allow a receiving unit to determine whether messages that it receives were generated by valid VSC units and have not been tampered during transit. Clearly, if an attacker could easily generate false messages or tamper with legitimate ones, then they could cause a receiving unit to have false beliefs.

In some cases, it is also necessary for a VSC receiver to be able to determine that a series of transmissions came from the same source. For instance, a receiver might want to determine the path of a given vehicle in order to assess road conditions.

2.4.2 Correctness

Even if we assume that we have some MI/OA service, there is still the possibility that a legitimate VSC unit will lie. Thus, we want to somehow ensure that the messages transmitted by a unit reflect the state of the world, at least as far as the transmitting unit knows.

2.4.3 Privacy

A secondary but still important goal of the system is privacy. In many cases, OBUs will be transmitting information about the state of the vehicle that could be used against the owner of the vehicle. For instance, if vehicles transmit their location, speed, and identity, it would be possible for terrorists to easily collect information about the movements of public officials by deploying sensors that listened for their OBU transmissions. This type of information leakage appears likely to make the VSC system unattractive to customers. Thus, it is not acceptable, for instance, to

issue every vehicle a Public Key Infrastructure (X.509) certificate [2] which is transmitted with each message.

The privacy requirement of our design, therefore, is that to the extent to which potentially compromising information is transmitted, it should be difficult to tie that information to any particular vehicle without being in close physical proximity to the vehicle (in which case the information would be self-evident). Thus, it is likely necessary to hide the identity of the transmitting unit.

It is not clear at the present time whether complete concealment of the transmitter's identity or merely plausible deniability is required. Note that in either case, the need for privacy interacts with that for MI/OA. For instance, it is sometimes necessary to be able to determine that messages had a common origin. This needs to be done in such a way that messages can only be linked over short time windows to avoid excessive information leakage.

Note that the goal of privacy is different from that of confidentiality. In most cases, the information being broadcast is inherently public, and it is just the identity of the transmitter that needs to be kept private. At this time, the VSC research has not identified any applications that require confidentiality. If those applications are identified, then that new requirement will need to be considered during the solution stage.

2.4.4 Robustness

Another secondary security goal is robustness. It must be assumed that some units will be compromised—either subverted into generating incorrect messages or have their keying material compromised. The system should have methods of recovering from that compromise. This will most likely require ways to allow legitimate units to ignore messages from compromised units.

Another important form of robustness is the ability to withstand *denial of service* (DoS) attacks. In such attacks, an attacker attempts to stop parts of the system from functioning. He might, for instance, jam all the radios in an area or attempt to send messages to individual units to stop them from receiving or transmitting. Such attacks are notoriously difficult to defend against. Our goal will be to defend against them where possible and, at worst to avoid making them any easier.

2.4.5 Fail-Safety

A related issue is that of failure modes. It may well be the case that a receiving unit has some reason to doubt the correctness of a message, though it is not entirely sure. It is not entirely clear how such messages should be handled.

For example, in many cryptographic systems, compromised units are added to a *revocation list*. Thus, when a unit receives a message, it checks the sender identity against the revocation list and rejects the message if the sender is on the list. Revocation lists are issued on some regular schedule (monthly is common). Now, consider what happens if a unit for some reason has not yet received the current revocation list. In theory, it then cannot validate any message because it cannot check for revocation status. In practice, if the unit has a relatively recent list -- according to which the sender is valid -- then with high probability the incoming message is still valid since attacks would be statistically rare. A system that failed safe would discard such messages if the list has surpassed its expiration date, but it would have substantially less information about the world.

2.5 A Strawman Security Architecture for Use in the Threat Model

In this section, we describe a strawman security architecture. It is not intended to be a complete design and indeed has known deficiencies with respect to the security goals we have just identified. It is intended to provide a common reference point for the rest of the threat model discussion and is necessary for that purpose only.

2.5.1 Providing Communication Security

In order to provide MI/OA, we divide the world into two categories: legitimate units and all others. All legitimate units will share a single symmetric key K_i , which is valid for time period i . Messages are transmitted with an attached *Message Authentication Code* (MAC), computed using K_i . i.e., the unit transmits message M as $M||MAC(K_i, M)$. Thus, a receiving unit can verify the correctness of a message via checking the MAC over M .

Each unit in the system will have a unique certified public/private key pair, but this key pair is used only for key distribution. Each group key K_i is issued by a single central authority, the *Key Distribution Center* (KDC), as shown in Figure 4. That authority only issues K_i to the units which are believed to be valid at the beginning of time period i . The KDC is usually operated by a *Security Officer* (SO) who is responsible for issuing keys and verifying that units have not been compromised.

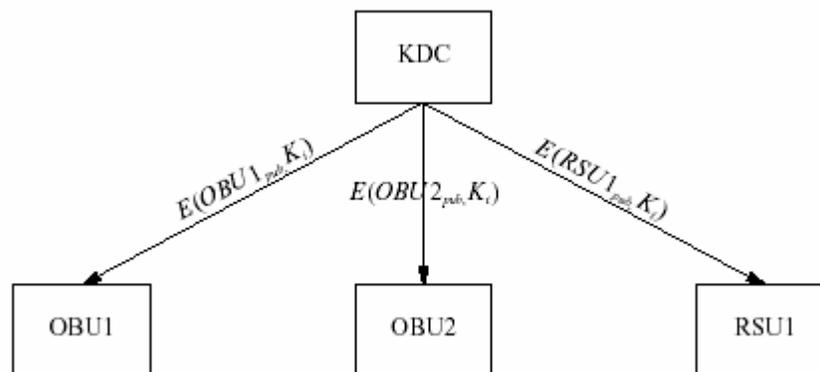


Figure 4: Strawman key distribution

It should be readily apparent that this system provides both MI/OA and privacy. As only valid units possess K_i , only they can generate valid messages and because the messages are MACed they cannot be changed in transmit. As all messages are MACed with the same common key, it is not possible to determine which messages were generated by which unit, thus privacy is provided.

In addition, this system provides a measure of robustness against compromise. Since the group key is periodically refreshed, if a unit is known to be compromised in time period i , the KDC simply does not issue it a key in time period $i+1$.

This strawman system is not a complete design or even necessarily the kind of security architecture that will eventually be deployed. It is provided purely as a reference point to focus discussion. The system just described has a number of known deficiencies. In particular, there is

no cryptographic way to differentiate RSUs from OBUs. Thus, an attacker who recovered the keying material from an OBU could impersonate an RSU. A hybrid system in which RSUs – which do not require privacy – are authenticated with public key cryptography could fix this problem.

2.5.2 Correctness

The system we have just described does not, however, guarantee correctness. An attacker in possession of a valid unit can extract K_i or the unit private key (through which he can obtain K_i) and can then transmit messages of his choice, posing as his unit. In order to prevent this attack, we need to harden the unit. The general idea is to enclose the sensitive portions in a tamper-resistant and tamper-evident casing, such as that specified in FIPS 140-2, levels 3-4 [3], and described below.

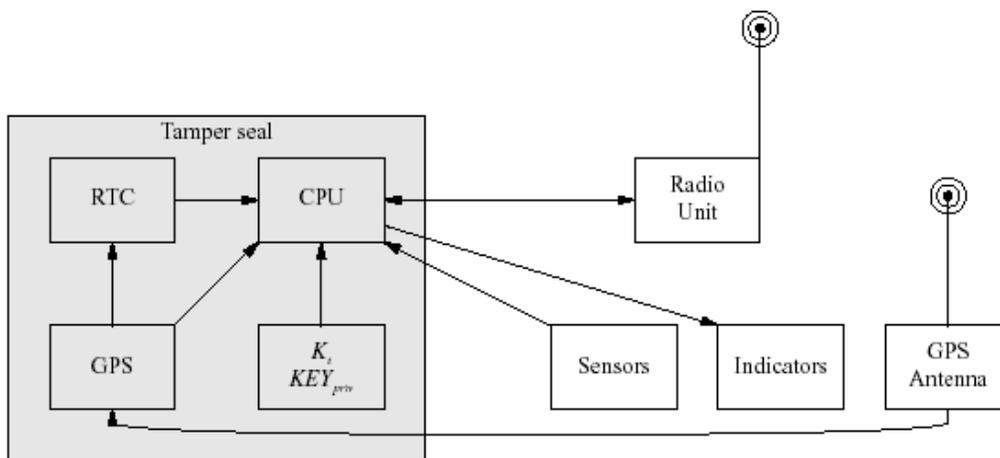


Figure 5: Strawman Block Diagram

The boundaries of the casing are an issue that will need to be decided during the full design process. For our strawman, we will consider a minimal configuration in which only the cryptographic unit, GPS receiver, a processor, and a real-time clock are embedded inside the tamper enclosure, as shown in Figure 5. The rest of the sensor modules lay outside the tamper enclosure. Therefore it is possible to convince the unit that, for instance, the user has applied the brake pedal by manipulating the appropriate sensor lead. In principle, one could place the GPS outside of the tamper enclosure as well; however, this would leave the unit with no way of independently determining the location of the unit and thus reduce correctness nearly to zero.

In general, there are two kinds of tamper seals: tamper-resistant and tamper-evident. A *tamper-resistant* seal is designed to be difficult to open and to ensure that the unit self-destructs (typically by zeroing the keying material) if an attacker attempts to open it. By contrast, a *tamper-evident* seal is simply designed to leave evidence that it has opened. Thus, if an attacker opened the unit and violated the tamper seal, it would be possible to subsequently determine this by physical inspection of the unit. For purposes of our strawman, we will assume that the seal is tamper-resistant, but that a sufficiently dedicated attacker might be able to extract the keying material.

2.6 Required Unit Capabilities

Now that we have covered each potential sample application scenario we can see that different types of units have different required capabilities. We may be able to exploit these differing requirements in our final design. We discuss those requirements here, beginning with OBUs.

2.6.1 OBUs

OBUs come in two varieties with different potential security requirements. They may either be intended for use in passenger vehicles operated by ordinary customers or they may be intended for public-safety vehicles such as ambulances and police vehicles. The requirements for these units differ both in terms of privacy and in terms of the desired degree of protection for keying material.

Protection of Keying Material

We expect that public-safety vehicles will need a higher degree of protection for their keying material. The rationale here is that public-safety vehicles will be able to create certain classes of messages that consumer vehicles cannot, e.g., "I'm an ambulance, out of the way." Clearly, forgery of such messages is especially undesirable and so public-safety OBUs need a higher degree of assurance. Some of this assurance will likely be provided by physical security around the vehicles themselves, but it is likely that the OBU will need better protection as well. Thus, public-safety OBUs might use better tamper sealing, be subject to periodic inspection, etc.

Note that in this scenario it would be desirable to somehow arrange that compromised consumer OBUs cannot be used to generate public-safety messages. The strawman system described in Section 3.5 cannot do this, because it requires the ability to verify a class of messages that the unit cannot generate. Generally, this requires public key technology, because with symmetric key, the ability to verify authenticity of a message is linked to the ability to generate valid messages.

Privacy

One goal of the overall VSC system design is to maintain privacy for traffic generated by customer OBUs. By contrast, public-safety vehicles generally do not require privacy. In fact, auditability for public safety is most likely desirable, especially if it allows easier revocation of compromised OBUs.

2.6.2 RSUs

RSUs also come in two varieties, as shown by the CSW and TSVW scenarios. The CSW RSU is a stationary unit that needs only to be able to generate the same message repeatedly.¹ By contrast the TSVW RSU is dynamically linked to the traffic signal and therefore needs a modest amount of intelligence.

Static Devices

A CSW RSU, in its simplest implementation, is a fairly dumb device and transmits the same (static) information repeatedly. Since the road it describes is stationary, it can be installed with information about the curve and never have to get any extra input. As a consequence, it does not need to have any sensors and is not vulnerable to sensor input-

¹ Note that a CS/RW RSU that adjusts its advice based on current weather conditions or observations of vehicle behavior is actually a dynamic RSU. Some applications, such as construction zone warnings, also might involve temporary movable dynamic RSUs.

based attacks. At most, it needs to be able to generate messages with new timestamps to avoid replay attacks. However it can be easily fitted with a real-time clock.

Similarly, a CSW RSU does not need to be mobile. It can always advertise the same position and curve data. In fact, it does not even need a GPS unit, since it can be programmed with its current position in an external fashion. (Although one might fit the RSU with a GPS as a fail-safe mechanism to ensure that it has been installed in the right location). For instance when the RSU is programmed with its curve speed information, it might also be programmed with its own position.

Dynamic Devices

The TSVW application scenario illustrates another kind of RSU, the dynamic RSU that transmits dynamic content. Unlike the CSW RSU, the TSVW RSU needs to know external states, in this case the traffic signal phase and timing information. It must, therefore, be able to generate some variety of messages containing different timings and phases. Like a CSW RSU, a TSVW RSU does not need to be mobile, since traffic signals are in fixed positions. However, one might imagine that some kinds of dynamic RSUs would move, e.g., construction work signals.

Common Limitations

Both kinds of RSUs share some limitations. Note that since the connection between any given RSU and its message is readily apparent, there is no requirement for RSU privacy. In addition, any given RSU only needs to be able to generate a fairly narrow class of messages. For instance, there is no need for a TSVW RSU to be able to generate messages containing maximum curve speed.

2.7 Attacker Capabilities

In the traditional *Internet Threat Model* (ITM), it is assumed that the attacker has complete control of the network but limited control of the end nodes. Rescorla and Korver write [4]:

The Internet environment has a fairly well understood threat model. In general, we assume that the end-systems engaging in a protocol exchange have not themselves been compromised. Protecting against an attack when one of the end-systems has been compromised is extraordinarily difficult. It is, however, possible to design protocols which minimize the extent of the damage done under these circumstances.

By contrast, we assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate. This means that the attacker can read any PDU (Protocol Data Unit) [message] on the network and undetectably remove, change, or inject forged packets [messages] onto the wire. This includes being able to generate packets that appear to be from a trusted machine. Thus, even if the end-system with which you wish to communicate is itself secure, the Internet environment provides no assurance that packets which claim to be from that system in fact are.

Unfortunately, this model of attacker capabilities is not completely applicable to the VSC case. In particular, VSC OBUs will be very widely distributed -- in the best case, one in every vehicle -- and so we must assume that some units will become compromised. Moreover, the trust issues involved in VSC are substantially more complicated than those of traditional e-commerce systems. In ordinary Internet systems, once one has generally authenticated the peer, one can establish which messages it is permitted to send and ignore others. In the VSC system, however, there are situations where authenticated peers can send messages that are potentially legitimate but actually malicious.

In addition, it may be necessary to reconsider the assumption of complete control of the network. This assumption is not entirely realistic in the Internet setting and is less realistic in the VSC setting due to the characteristics of wireless data transmission. Depending on the security service that we are attempting to achieve, this may be an asset or a liability.

Finally, we need to consider the likelihood that the VSC system will face multiple attackers with varying capabilities. Rather than design a system that must be immune to all attackers, we shall focus on designing a system that intentionally presents different levels of security to attackers with different capabilities.

We consider four types of attackers, roughly in order of increasing capability:

1. Attackers with a programmable radio transmitter/receiver;
2. Attackers with access to an un-modified VSC unit (with an intact tamper seal), either OBU or RSU;
3. Attackers who have access to a modified VSC unit and who have obtained the keying material; and
4. "Inside" attackers who have access to records and equipment operated by the vehicle manufacturer or the VSC unit manufacturer.

In the rest of this document, we will be referring to these attackers as "Class 1", "Class 2", etc.

In the next four sections we describe each type of attacker and the type of attacks they might be able to mount. Note that we explicitly assume that attacks on the GPS satellites and base stations are outside the scope of this task. If an attacker can compromise GPS, the consequences will extend far past those on this task.

2.7.1 Class 1: Attackers with Programmable Radio Transmitters/Receivers

The most limited kind of an attacker is one with a programmable radio transmitter/receiver. This might be a modified 802.11 radio, a hand-built hardware device, or a software-defined radio. In any case, we assume that such an attacker can receive all VSC messages in his area as well as generate any number of legitimate-appearing VSC messages. He will also be able to broadcast with effectively unbounded signal power.

Class 1 attackers do not, however, have access to any VSC keying material. Thus, essentially any communications security measures whatsoever will keep them from mounting forgery attacks. However, there are still a number of attacks they could potentially mount.

Replay/Tunneling

Although a Class 1 attacker cannot craft or modify messages, they can quite easily arrange for them to appear at a different point, either in time or space. Thus, an attacker can store a message and retransmit it at a later time, amplify a message so that it appears in a distant location, or both. Protecting against this attack clearly requires that receiving units have a verifiably correct idea of where they are and what the current time is and use that to reject expired or irrelevant messages.

Denial of Service

The most straightforward attack for a Class 1 attacker to mount is a DoS attack. The simplest kind of DoS attack for such an attacker is simple radio jamming. Since a Class 1 attacker can have a network transmitter of essentially unlimited power, it is fairly straightforward to prevent all transmissions from being received. There are also a number of more sophisticated DoS attacks on 802.11 [5].

In addition, it is very easy to perform "selective jamming" on an 802.11 system: the attacker listens to enough of the message to know if he wants to let it go though. If not, he transmits during the 2nd part of the message transmission. In this way, he controls what part of the overall state gets through.

There are a large number of other potential DoS attacks. For instance, one could simply flood the network with transmissions. This will not only crowd out other valid transmissions but also potentially consume prohibitive amounts of CPU time on the receiving side as the receivers attempt to verify the transmission's validity. Finally, one could simply physically damage a unit so that it couldn't provide any service whatsoever.

RF Fingerprinting

Even a Class 1 attacker can mount a substantial privacy attack by means of *RF fingerprinting* [6]. Even in mass-produced units, radios have characteristic emissions profiles. It is generally possible to distinguish one radio transmitter from another by use of these emission profiles. Thus, an attacker who can observe the emissions of a given vehicle can then subsequently identify that vehicle by its transmissions alone. We will investigate the difficulty of this attack before the solution phase.

Remote Compromise

Even the best security implementations occasionally contain programming flaws. If those flaws are in the early message verification and processing code, then a Class 1 attacker might be able to compromise the security implementation and thus the VSC unit. Note that such an attacker could presumably only exploit code in the security layer, since it cannot generate messages that will pass the initial security checks.

A remote compromise of this type can often be used to gain control of the entire unit. In such a scenario, the attacker would effectively be a class 3 attacker, but without needing to break the tamper seal.

Remote Management

It is expected that in some cases RSUs will be remotely manageable, either via the Internet or wireless channels. In such cases, there is obviously a concern that an attacker will send false management messages. This sort of remote authorization and access control is a standard security problem with many well-known solutions. The most appropriate solution will need to be determined during the solutions phase.

2.7.2 Class 2: Attackers with Access to an Unmodified VSC Unit

Unless measures are taken to control access, it will in general be very easy for attackers to obtain VSC units. Most likely the attacker can obtain an arbitrary number of OBUs from junked vehicles but, in the most expensive case for the attacker, they can simply buy them along with a used vehicle. Initially, the price per VSC-enabled vehicle will be high (because they are newer), but the price will come down quickly if VSC units are available in most new vehicles.

Obtaining RSUs will probably be even easier, since it is assumed that they will be embedded in unattended roadside devices. Thus, our threat model needs to assume that an attacker will be able to obtain either an OBU or an RSU. In this section we consider the capabilities of a Class 2 attacker, who has a VSC unit but has not broken the tamper seals in order to compromise the CPU or keying material.

Change of Location

The most straightforward attack on an RSU is simply to move it. One might, for instance, take a stop sign transmitter and move it to another location, where it forces people to stop. This has a similar result to the tunneling attack described above, but is executed differently. It could potentially be combined with the indicator mismatch attack, described below.

Indicator Mismatch

A number of the infrastructure-to-vehicle services are designed to inform the user about the state of some piece of infrastructure. For instance, the TSVW application scenario is intended to represent the state of a traffic signal. Even if the VSC unit itself is undamaged, it is trivial for the attacker to change the state of the signal itself. For instance, one could rewire the traffic signal so it believes it is emitting a red signal but really is emitting green.

The danger here is that users of VSC might come to rely upon VSC to get information about the state of the world. To the extent that there are multiple indicators, some VSC-based and some legacy, there is a potential for damaging misunderstandings. Consider, for instance, the following scenario, shown below in Figure 6.

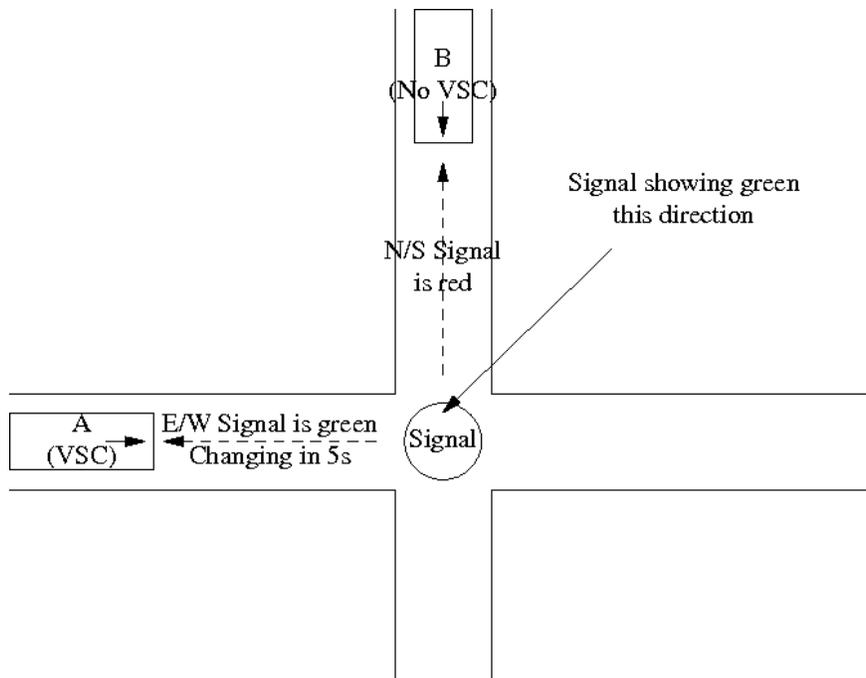


Figure 6: Signal/Message Conflict

In Figure 6, Vehicle A is VSC-enabled and Vehicle B is not. The attacker has tampered with the traffic signal so that the RSU and the actual signal are out of phase. Thus, B has a green signal and A does not see a TSVW warning. If A has a careless driver, he might rely on the fact that he has not seen the warning and not notice that the signal was in fact red. There is thus a potential for a collision. It is worth noting that this sort of scenario

can be created even without VSC, simply by wiring both indicators to be green. However, to the extent to which users trust VSC and their behavior becomes more aggressive due to the fact that they know it is active, we might expect that the problem would be worse in the case of this attack. In addition, if this attack is mounted by making both lights green, this is immediately and visibly obvious, whereas it is less obvious to detect when the VSC and visual signals conflict.

Sensor Spoofing

In our strawman design, we assume that the physical sensors (accelerometers, brake sensors, radar, etc.) are outside the tamper boundary. As these sensors often need to be in specific, physical locations this seems very likely. However, this opens up a variety of possible attacks. Provided that the attacker can generate the correct sensor inputs, he can use the VSC unit as an oracle to generate whatever false messages he desires.

A simple example of an attack using this technique can be found in the EBL service. The attacker modifies his vehicle to spoof the accelerometer and brake pedal sensors, and drives along the highway. He then waits until he is in front of the victim and sends braking signals to the OBU. The OBU, in turn, generates a valid emergency braking message over the air interface. The OBU behind the attacker receives the message, concludes it needs to brake, and informs the driver.

GPS Spoofing

Both OBUs and RSUs are potentially susceptible to a GPS-spoofing attack. Although we have assumed that the GPS is within the tamper boundary, GPS receivers rely upon radio transmissions to determine their position. With sufficient effort it may be possible to convince the GPS receiver that it is in an arbitrary location at an arbitrary time simply by supplying the correct signals to its antenna input. We will investigate the difficulty of this attack before the solution phase.

If the attacker can use the GPS to convince the VSC unit that it is in a different location or at a different time, then he can use it to generate an arbitrary number of messages with a time and date stamp of his choice. There are a large number of attacks that can be mounted via this mechanism. GPS spoofing attacks are particularly damaging when combined with sensor spoofing attacks. With GPS spoofing, an attacker can mount the fake braking attack described above but without even being in his vehicle--or with a powerful enough transmitter, even in the area.

2.7.3 Class 3: Attackers Who Have Recovered Keying Material From a VSC Unit

No matter how good a tamper seal is attached to the VSC unit, a sufficiently dedicated attacker is likely to be able to extract the keying material. In addition, an attacker might be able to exploit bugs inside the VSC unit code to compromise the unit without violating the tamper.²

Accordingly, we need to consider attacks in which the attacker has compromised a VSC unit and recovered the private key and/or whatever temporal keying material is inside.

² It is not currently known how to extract the keying material of FIPS 140-1 level 4 certified tamper-resistant devices. However, it is expected that it is technically possible. Moreover, it is likely that level 4 protection will be cost prohibitive for VSC.

In general, a Class 3 attacker can mount any attack that a Class 2 attacker can mount.³ In addition, a Class 3 attacker can generate essentially any message that the VSC unit could have generated.

Duplication

The most important capability that a Class 3 attacker has that a Class 2 attacker does not is that he is no longer subject to the constraint that he needs the VSC unit. Thus, he can easily build any number of compromised units for the one time cost of breaking a single VSC unit.

Physical Law Violations

Another capability that a Class 3 attacker has that a Class 2 attacker does not is the ability to generate completely incoherent message sets. For instance, a VSC unit can be programmed to know that time cannot run backwards, whatever the GPS input says. Similarly, the VSC unit can be programmed to know that position changes that exceed 500 mph are physically impossible and reject such input. Thus, one could conceivably program the unit with implausibility checks to refuse to generate messages that imply implausibly high velocities. The details of such filters should be investigated in future VSC-related work. By contrast, because a Class 3 attacker has access to all of the keying material, he can generate any messages he chooses, including those with implausible physical counterparts.

Increased Ability to Compromise

Another consequence of the Class 3 attacker's ability to generate any message is an increased ability to exploit programming errors in other VSC implementations. We expect that the first check that a VSC implementation will do upon receiving a message is to verify the MI/OA properties. Since other attackers have quite limited message sets they can generate, the chance that they can exploit bugs in the message processing code is fairly low. They must focus their attacks on the cryptographic code. By contrast a Class 3 attacker can always generate a message that will pass the MI/OA check and thereby exploit bugs in the message processing code.

2.7.4 Class 4: 'Inside' Attackers

The final class of attackers is referred to as "Insiders." VSC units need to be manufactured, and an attacker who is employed by the OBU/RSU manufacturer would have substantial leverage to mount additional attacks. There are actually two groups of Class 4 attackers: authorized and unauthorized. An authorized attacker would be one who was operating with the consent of the manufacturer. An unauthorized attacker would likely be a disgruntled employee. Since the capabilities afforded to an authorized insider are a superset of those offered to an unauthorized insider, we will focus our analysis on authorized insiders.

The primary additional capability that an inside attacker has is access to the KDC and the customer records. He can potentially use this to extract keying material or get a list of all units, their owners, and whatever identifying information is known. This allows him to mount a number of attacks.

³ One exception here is that one could imagine placing the radio inside the tamper boundary and hope that when the tamper is violated it disables the radio or at least alters its fingerprint. In such a case, a Class 3 attacker could not mount Class 2 attacks if RF fingerprinting is used as part of the authentication system. We will not fully consider this issue here, since placing the radio inside the tamper is probably prohibitively expensive, and it is doubtful that fingerprinting technology is cost-effective to be used in every receiver.

Key Extraction

In the strawman scheme of Section 3.5, an attacker who controlled the KDC would likely be able to obtain a copy of the current group key K_i . Such an attacker would effectively have the capabilities of a Class 3 attacker. Because this sort of attack is not in the interest of the manufacturer, we assume that this sort of attack is primarily limited to unauthorized attackers. Similarly, an attacker who purchases keying material from an insider can become a Class 3 attacker.

Cryptographic Attacks on Privacy

If the OBU/RSU manufacturer maintains a database of all units manufactured and their public keys, then an authorized attacker is in a position to mount a number of attacks. In the strawman scheme described in Section 3.5, the scope of these attacks is somewhat limited. Since all units use the same group key, access to the keying material does not allow unit identification. However, the attacker could still determine a user's rough location when they request a new K_i . Depending on how often K_i is changed, this might provide fairly high-resolution information.

Some other cryptographic schemes provide much more information leakage. For instance, one commonly proposed scheme is Group Signatures [7,8]. In such schemes, there is an escrow agent that can map any signature to the private key that signed it. With such a scheme and a key-VIN database, a Class 4 attacker could potentially identify any entity.

RF Fingerprinting Database

If the manufacturer maintains an RF fingerprint of each unit, a Class 4 attacker would potentially be able to identify any unit by its fingerprint, as opposed to just tracing a given fingerprint, as a Class 1 attacker could. A database mapping fingerprint-to-VIN would be ideal for this purpose. The appropriate level of protection against this form of fingerprinting attack will need to be determined during the solutions phase.

Attacker Type	Attack
Class 1	Replay/tunneling Denial of service RF fingerprinting Remote compromise Remote management
Class 2	Change of location Indicator mismatch Sensor spoofing GPS spoofing
Class 3	Duplication Physical law violations Increased ability to compromise other units

Class 4	Key extraction Cryptographic attacks on privacy RF fingerprinting (with database)
---------	---

Table 1: Summary of attacker types and attacks

2.7.5 Some Out Of Scope Threats

There are a number of attacks on the system that are out of scope for this task. These are described below.

Physical DoS Attacks

It is clearly possible to mount a DoS attack on any OBU or RSU simply by depriving it of power, physically damaging it, destroying it, etc. The solution to this kind of problem is physical hardening. We will not address this kind of problem here.

Radio Jamming

As discussed in Section 3.7.1, it is quite possible to build a radio transmitter that will make the entire 802.11 network-- or specific units--unusable. Since we rely on the 802.11 network, these attacks are out of scope for us.

Attacks on the GPS System Infrastructure

An attacker who could compromise the GPS satellites or base stations could easily change any unit's idea of position. We do not address security for the GPS system. At the present time, there is no way of authenticating GPS messages. However, we expect that units will need some set of heuristics to assist in detecting false GPS information. During the solutions phase we will attempt to determine some such heuristics.

Software-Based Compromise of Units

The OBUs and RSUs will contain software that processes network traffic. In many cases it is possible to mount attacks on such software that allows one to take over such systems. We advise that secure programming practices, such as those described by Viega and McGraw [9], be used when developing VSC units. As this kind of protection is a software issue and not a communications issue, it is out of scope for this task. Containing compromise of such units is, however, in scope.

Misconfiguration (Accidental or Intentional)

In many cases, RSUs must be configured when they are installed. For instance, a CSW RSU must be programmed with the characteristics of the curve that it is providing information for. In some such cases, the personnel in charge of inputting that information will make errors. Intentional misuse is also possible and should be guarded against. However, defenses against all such attacks are out of scope for this task.

2.8 Containing Compromise

Finally, we turn to the issue of dealing with compromised units. There are two basic strategies for containing compromise:

- Proactively limit the capabilities of a unit in order to minimize the effect of compromise.

- Reactively contain a compromise once it has occurred.

We consider each approach in turn. The purpose of this section is not to describe solutions but merely to explore the potential parameters of a solution and the implications of various design choices.

2.8.1 Proactive Containment

The basic principle of proactive containment is to limit capabilities. As described in Section 3.6, not all VSC units need to have the same capabilities. Since every additional capability that a unit has is a new capability that an attacker might employ, limiting unit capabilities proactively contains potential compromises. For our purposes, the two most important role separations are those of unit type and mobility.

2.8.2 Discriminators for Compromised Units

The reactive way to contain compromise is to identify and isolate compromised units. Strictly speaking, what we wish to achieve is not to *identify* compromised units but rather to isolate them. Thus, it is not necessary to identify which units have been compromised but merely to have a way to reject messages from such units in the future. Thus, what we concern ourselves with here is obtaining a *discriminator* – a piece of information which is sufficient to reject future traffic from such units. In general, however, it seems likely that such a discriminator will in fact be the unit's identity.

The problem, then, is for the Security Officer to derive a discriminator. We consider several types of information which one could potentially use for this purpose:

- Some set of messages that are deemed to be bad (apparently valid, but incorrect).
- The identity of the compromised unit (e.g., a serial number).
- The revealed private key of the compromised unit.
- The compromised unit itself, if one has physical possession of it.

Once the SO has derived a discriminator, the next stage is to somehow contain the compromised unit. There are a number of possibilities, ranging from simply telling the compromised unit to shut down (which only works with Class 2 attackers) to compiling compromised unit lists to denying the compromised units keying material. These options will be fully explored during the solution phase.

2.8.3 Residual Capabilities of Compromised Units

Our objective for containing compromised units is primarily to stop them from transmitting false information. However, compromised units will still be able to receive transmissions. As the current threat model does not include confidentiality, such attackers will have essentially the same capabilities as any Class 1 attacker.

3 Constraints

3.1 Introduction

This was the second step in the security effort for the VSC project. The first step described the VSC threat model: the capabilities of potential attackers and the types of attacks such an attacker might be able to mount. This section describes the external constraints that a proposed solution must take into account.

Any engineering project is a compromise between the desirable features of the design and the constraints on it. The threat model enumerates the desired goals. This section enumerates the constraints on the design. The primary concern when introducing security is that it imposes additional overhead, for example:

- cryptographically secured messages are larger than ordinary messages
- hardened security modules are larger than ordinary computers and may consume more power
- dedicated security processors are likely to add cost to units
- security systems require management which may be expensive

A security system design must take all of the related boundary conditions into consideration if it is to be successful, or it runs the risk of being un-deployable. Note that the constraints listed below are the most accurate that the security team was able to determine, though in some cases they are accurate only to an order of magnitude. The constraints described in this section represent boundary conditions only, and must not be interpreted as specifications. They are accurate enough for this security study, but they are not meant to be used for any other purpose.

Network characteristics – The VSC security solution must operate in a network environment defined by the DSRC protocols. The solution must operate within the communications and timing parameters provided by that network.

Environmental characteristics – The VSC security solution is designed for deployment either in vehicles or as part of infrastructure devices. These environments present constraints in terms of power consumption, heat, form factor, etc.

Cost of goods – VSC units, whether *On-Board Units* (OBUs), *Road Side Units* (RSUs), or *Management Devices* (MDs) must be manufactured. The cost of goods must be kept within certain acceptable limits.

Management costs — The final VSC system will require management and maintenance, both of the system itself and of the deployed devices. Again, these costs must be kept within acceptable limits.

In the remainder of this section, we consider each form of constraint. In Section 5, which illustrates a proposed VSC architecture, we will explore options for producing a system that provides the security guarantees described in Section 3 while still meeting as many of the constraints described here as is practical.

3.2 Network Characteristics

Vehicle safety communications will be operating over an 802.11-based ad hoc wireless network. The network operates at variable data rates, ranging from 3 Mb/s to 27 Mb/s. The range of the network is approximately 300 meters in light traffic situations and 800-1000 meters without traffic or obstructions.

3.2.1 Total Number of Units

Security management overhead can depend on the number of total units in the VSC system. For our purposes (and our purposes only), we estimate that there will be on the order of 10^9 units in the United States (four times the number of vehicles in the US today) and 10^{10} units in the world as a whole (about 1.5 times the world population today). These are, of course, very approximate numbers. They are best-guesses for the order-of-magnitude number of units, and they are accurate enough for the purposes of this security task. Since the system design must consider long-term interoperability, the security system must assume full deployment, even if this constitutes overkill at first deployment; one security system for low penetrations to then be retrofitted as the deployed population grows would be difficult and/or unmanageable.

3.2.2 Maximum Data Rate

The most basic constraint that we must accept is the data rate of the channel. This presents a hard upper bound on the amount of data that must be processed by a receiving VSC unit. We assumed a maximum data rate of 6 Mb/s, which is the nominal rate of the "control channel" over which data related to safety applications will be flowing. The system should be able to handle bursts of 27 Mb/s, but need not necessarily process it all in real time.

3.2.3 Packets Per Second

An additional boundary is the number of packets per second that an implementation must have the potential to verify. Given the above network constraints, we estimate that this number is 4000 packets per second (pps) for a data rate of 27 Mb/s.[10] At 6 Mb/s, the packet rate is approximately 2500 pps. Note this need not necessarily happen in real time. Low priority messages may be handled out of order.

3.2.4 Latency

A critical question is that of latency. In order to be useful, application messages need to be delivered within a reasonably short time after they are transmitted. The exact time varies by application. Based on protocol research thus far, we assume a maximum allowable latency of 100 ms, which would include all of the time between when the data to be sent from the sending vehicle is collected and when the receiving vehicle has an opportunity to react to the received data.

However, the communications system itself introduces some latency. Thus, the total amount of latency introduced by the security layer must be less than the total allowable latency minus the communications latency. Based on results from the VSC protocol research activities, we estimate that communications latency is approximately 10 ms, and that we should conservatively design the security layer to have equivalent latency. Note that this is total latency, including both data transmission costs and data verification costs.

There may be applications developed in the future that could benefit from the VSC units' ability to route messages. In the instance of stationary RSUs, some RSUs may be physically connected to a network by wire, or occasionally through a management console, in order to receive information updates. This RSU could forward information to other RSUs within transmission range and request that the other RSU forward messages to yet another set of RSUs. In this instance, the RSU may know *a priori* which RSUs are its neighbors or will have sufficient time to discover them through some neighborhood discovery algorithm. The latency introduced during discovery and maintenance of routes in this RSU-RSU routing scenario should not impose constraints. However, packet forwarding does impose latency. We are assuming that no time-critical applications will require RSU packet forwarding.

In the case of routing among stationary OBUs, applications may include the location of a vehicle in a large airport parking lot, finding a parking spot in a crowded city, etc. The main issue is that in stationary OBU-OBU routing, the latency introduced by neighborhood discovery and maintenance again must not create a problem. We are assuming that no time-critical applications will require packet forwarding by OBUs or of OBU messages.

3.2.5 Packet Loss Rate

DSRC provides a broadcast channel with no guarantee of delivery. Therefore, we must assume that some packets will be lost. Based on simulation results so far, a broadcast's effectiveness ranges from 80% to 50% for a range from nearby to 100 meters away, under stressful but not excessive channel conditions. Of course, these numbers are highly dependent on the actual parameters in the simulation, such as transmission power, packet size, vehicle density, etc. In general, it is unlikely for a broadcast to be more than 90% successful at the intended communication range under stressful load. Conservatively, we will assume an 80% packet loss rate.

3.2.6 Packet Size

In order to maximize bandwidth usage and reduce processing overhead, we need to ensure that the security layer does not add excessive overhead to packets. Based on simulations run during the VSC protocol research activities, we assume final packet size no larger than 200 bytes for most packets, with at least 100 bytes of that used for application data. Therefore, the target is to have no more than 100 bytes of security information carried in most packets. Note that this does not mean that packets larger than 200 bytes can never be used. Larger packet sizes may be valuable for setup and management purposes. However, the typical packet size for safety messages should be less than 200 bytes.

Constraint type	Constraint value
Aggregate bandwidth	6 Mb/s
Maximum received packets/second	4000
Maximum allowable latency	100 ms
Maximum network latency	10 ms
Maximum packet size	200 bytes

Table 2: Summary of Network Constraints

3.3 Environmental Characteristics

The environments in which OBUs and RSUs must be deployed impose certain physical constraints. For instance, OBUs must be small enough to fit into a vehicle, must not draw too much power, etc. This section describes some potential environmental constraints for the purposes of this security study. Individual end-user requirements may differ. In general, these constraints will be different for OBUs and RSUs.

3.3.1 Unit Size

RSU

RSUs are embedded in a variety of devices. The smallest likely such device is a road sign or traffic signal. Thus, RSUs should be able to fit in a form factor approximately 30 cm x 20 cm x 10 cm. This restriction applies to the entire VSC device, including security but not including antennas. We anticipate that RSUs will be able to mount substantial antennas in order to adapt to their installation environment.

OBU

As OBUs are fitted inside the vehicle, they need to be relatively small. OBUs should be able to fit in a volume of approximately 500 cm³. It is unlikely that they will need to be thinner than 1.5 cm. This restriction applies to the entire VSC device, including security but not including antennas.

3.3.2 Temperature Range

RSU

RSUs need to be safely deployable in the most extreme weather conditions found in roadside environments. Therefore, they should be operable in the range -40 to +85 C.

OBU

OBUs need to meet the same temperature tolerance standards as other pieces of vehicle electronics and instrumentation. In general, this is -40 to +125 C. If necessary, it may be possible to place the OBU in a protected environment, in which case the operating range would be approximately -20 to +70 C.

3.3.3 Heat Output

RSU

RSUs can typically be placed in ventilated environments. Therefore, the practical limits on RSU heat evolution are those for general purpose computers. We have somewhat arbitrarily chosen a target of 50W.

OBU

OBUs need to meet the same heat output standards as other pieces of vehicle electronics and instrumentation, which is on the order of 10W.

3.3.4 Power Consumption

RSU

RSUs generally draw less than 100 W. The peripheral equipment (e.g., sensors) usually draws far more power than the radio unit, according to industry representatives from the DSRC standardization group. As solar powered units are generally installed where plenty of sun is available, they can be assumed to have plenty of power. We have somewhat arbitrarily chosen a target of 50 W for RSUs.

OBU

OBUs must run off the vehicle battery and alternator, like any other electronic device in the vehicle. Unfortunately, acceptable power draws vary widely. We are assuming the following parameters as typical, though of course each automaker has independent

specifications. The numbers below are used only as boundary conditions and only for the purposes of this security study.

Max current draw		30-100 mA
Typical current draw	7	0% of max
Sleep mode current draw (vehicle off)		<1 mA
Supplied voltage (vehicle off)		12 V
Supplied voltage (vehicle on)		14 V
Operating voltage		9-16 V

Constraint	RSU value	OBU value
Unit size (cc)	6000	500
Temperature range (degrees C)	-40 - +85	-40 - +125
Heat output (watts)	unlimited	10
Power consumption	unlimited	30-100 mA at 14 V

Table 3: Summary of Environmental Constraints

3.4 Cost of Goods

The various devices in the system must be manufactured. The cost of these devices must be factored into the overall cost of the units in which they are embedded. This section discusses assumptions regarding those costs for the purposes of this security study only.

3.4.1 RSUs

Without security, road-side units used for currently-deployed systems such as toll collection and traffic monitoring cost about \$10,000-20,000. According to members of the DSRC standards writing group, efforts are currently being made to bring the price down to approximately \$200-\$500 (\$1000 for high-end units). Security should add less than \$200 to the cost of the unit.

3.4.2 OBUs

The maximum acceptable cost of an OBU is assumed to be approximately \$100, and security should add less than \$50 to the cost of an OBU. Preferably, the additional cost would be less than \$25.

3.4.3 Management Consoles

RSUs are expected to be field-manageable and perhaps remotely manageable. Field and office personnel will need to be issued management consoles. These management consoles need to be able to securely communicate with RSUs. The maximum cost of the management consoles should be less than \$10,000.

3.4.4 Service Consoles

In some cases it may be necessary for service personnel such as auto dealers to maintain OBUs. Currently, service personnel are required to purchase substantial tooling from the manufacturers,

at costs of up to \$10,000. Therefore, we believe that such OBU maintenance stations could cost up to \$10,000.

3.4.5 Manufacturing Facilities

In order to avoid the construction of fake OBUs, OBUs must be manufactured in secure facilities. The security requirements for those facilities should not increase the fixed cost of facilities by more than 25 percent.

3.5 Management Costs

Any large-scale security system requires some management and this system is no exception. These management costs are borne by a variety of parties, ranging from OEMs to transit agencies. This section discusses assumptions regarding management costs for the purpose of this security study only.

3.5.1 Investigation of Compromise

Containment of compromised units is a design goal of the system. When such a compromise occurs, it is first necessary to derive a discriminator for the compromised unit. Such a discriminator can then be used to shut the compromised unit out of the system. The likely scenario is that a unit is compromised, and then the attacker uses the compromised unit maliciously. An investigation must then be mounted to derive the discriminator.

The potential cost for such an investigation is naturally dependent on the size of the attack. For a major accident, we estimate the cost to be on the order of \$20,000 for the VSC portion of the investigation (and misuse of VSC would have to be suspected for it to be investigated). For traffic disruptions, we roughly estimate that if a disruption is suspected of being caused by malicious use of VSC, the cost of the investigation would be less, perhaps on the order of \$5,000.

3.5.2 Key Infrastructure

OBUs are likely created with their keying material installed, making management relatively simple. However, in the proposed model, RSUs generally will be using a conventional public key infrastructure. Thus, a hierarchy of authorities will be required to certify RSUs as well as other certificate authorities. The distribution of these costs depends on the PKI architecture.

The topology of the PKI impacts on efficiency and ease of deployment. The PKI is primarily responsible for the issuance of digital certificates and revocation information. Certificate Revocation Lists (CRLs) are the most common form of revocation information. For RSUs, the naming structure is assumed to be hierarchical, based on geopolitical areas. For example, an RSU might have a name like *C=US, S=VA, L=Fairfax County, CN=RSU 10048*. A name like this can be accommodated under several different PKI topologies. We need to consider two major alternative architectures *hierarchical* and *flat*.

Both Public Key Infrastructure topology options are described below.

Hierarchical PKI

The PKI topology mirrors the levels in the naming hierarchy. For example, the U.S. DOT might operate a CA for the top level, issuing certificates to each of the States. Then, the States operate CAs, and they issue certificates to each of the counties and cities within the State borders. Finally, the local governments operate CAs, and they issue certificates to the RSUs.

This alternative appears to directly match the expectations of most people familiar with governmental structures. A single trust anchor is needed to represent each country. A

superior international organization could be created to serve as the top-level CA, but it would impose additional efficiency concerns.

This hierarchal structure imposes a burden on the OBU. Using the above example, four signature validations are needed to verify a message from RSU 10048. They are:

1. Validate the signature on the VA certificate with the U.S. DOT public key.
2. Validate the signature on the Fairfax County certificate with the Virginia public key.
3. Validate the signature on the RSU certificate with the Fairfax County public key.
4. Validate the signature on the message with the RSU public key.

Further, revocation information provided by each of the CA ought to be checked. If CRLs are employed, three addition signature validations are needed. Each of the CRLs ought to be small since each CA is not responsible for a large number of certificates.

Deployment requires each of the superior organizations to be operational prior to a subordinate governmental entity. Therefore, Fairfax County cannot install their first RSU until the U.S. DOT and the Commonwealth of Virginia are both fully operational.

Flat PKI

A single CA issues all of the certificates within a nation, but the naming hierarchy is unchanged. For example, the U.S. DOT operates the CA for the top level, but it accepts requests for certificates and revocations from authorized registration authorities (RAs) in each of the States, counties and cities. The process for authorizing RAs will likely follow the natural governmental structure, but once authorized, the RA can communicate directly with the CA.

All of the management concerns are transferred to the RA registration process. And, the CA is responsible for ensuring that one local government cannot request certificates that are affiliated with other jurisdictions.

This flat structure reduces the burden on the OBU. Using the above example, two signature validations are needed to verify a message from RSU 10048. They are:

1. Validate the signature on the RSU certificate with the CA public key.
2. Validate the signature on the message with the RSU public key.

Revocation checking can be problematic in this structure, since all of the revoked certificates are listed in a single CRL. While only a single signature is needed to validate the CRL, the number of entries on that CRL can become quite large. A large CRL takes significant bandwidth to download. This concern might be mitigated by performing the download at gas stations where the vehicle is stationary.

Deployment only requires the national CA to be operational. Therefore, Fairfax County cannot install their first RSU before the Commonwealth of Virginia has installed any capabilities, assuming that the registration process can be accomplished by paperwork.

This alternative has an additional concern. The national CA must be highly available. In the first alternative, the outage of any single CA has very little impact on the overall system, but in this alternative, an outage in the national CA can have major ramifications. Therefore, an architecture that includes concurrent operations from multiple physical locations is needed, making the national CA more expensive in this alternative.

3.5.3 Management of RSUs

Unlike OBUs, RSUs require substantial management. For instance, a transportation agency might want to change the settings on a curve speed warning RSU in order to accommodate changing road conditions. This sort of remote management incurs costs. In general, management will be performed by personnel with minimal computing background training, and it must be performed quickly. Reprogramming an RSU with new parameters should be possible in less than 30 minutes.

4 Architecture

4.1 Introduction

The previous two sections described the VSC threat model and system constraint assumptions. This section describes some of the possible architectural options for the VSC Security System. A full protocol based on these architectural elements is described later in Section 6 of this report.

This chapter describes the high-level security architecture that was developed to address as many of the threats identified in Activity 2 as possible, while still fitting within the constraints identified in Activity 3. The purpose of this security architecture development was to create a “cryptographic skeleton” design, showing the various communicating parties, the message flows, and the cryptographic transforms to be used.

4.2 Architecture Summary

The different privacy and security assumptions of RSUs/PSOBUs and OBUs result in different security strategies. A standard PKI strategy for RSUs/PSOBUs was assumed, and a variety of strategies for providing authentication for OBU messages were described. After much consideration, Anonymous Certificates were identified as the most promising OBU authentication strategy.

Security that meets the requirements described in the threat model imposes significant constraints. With the currently proposed architecture, which includes many optimizations, each message transmitted would include significant overhead, and the message signatures would take time to process once they are received. Management of a Public Key Infrastructure for RSUs would also be necessary, according to the proposed scheme.

Two major decisions are:

1. What PKI structure to use for RSU authentication, and
2. Which signature algorithm to use for message authentication.

ECDSA has been recommended, but it is not the only alternative. The protocol design proceeded assuming that ECDSA will be chosen, but with sufficient flexibility to allow future changes.

4.3 Architectural Overview

This section provides an overview of a proposed security architecture, which is then described in detail in subsequent sections.

4.3.1 Key Hierarchy

In order to preserve anonymity for OBUs while still allowing for fine-grained authority control for RSUs, VSC Security could use a dual authentication structure as shown in the figure below. Under this model, consumer OBUs would be authenticated via one set of mechanisms and all other units would be authenticated via another set of mechanisms. This is one possible answer proposed for the purposes of studying the implications of the model and its associated assumptions.

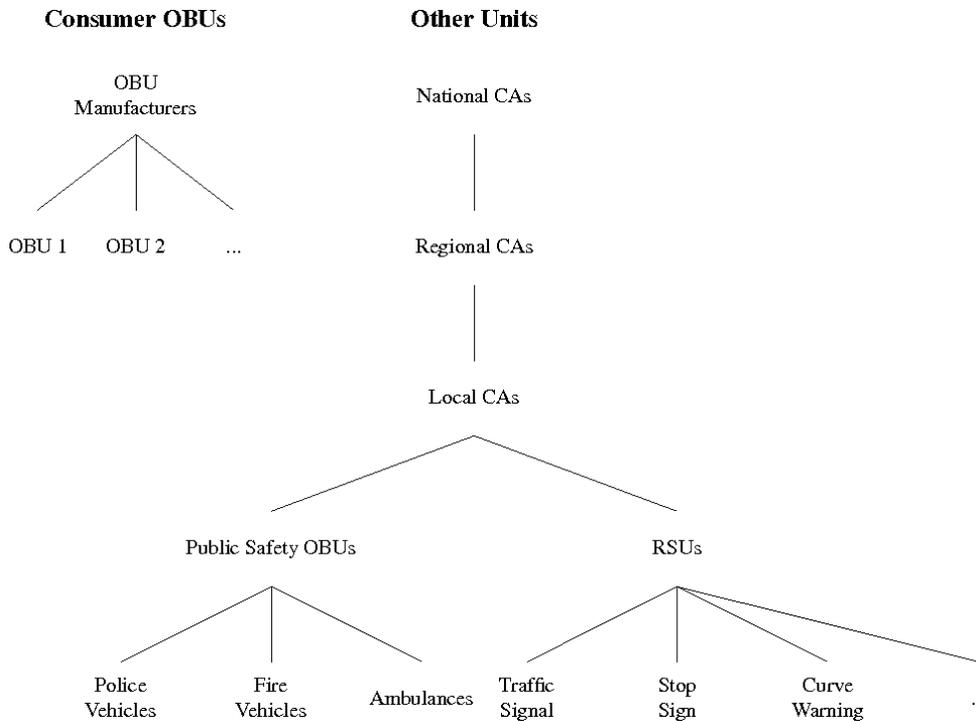


Figure 7: Overview of Dual Authentication Structure

The figure above shows the proposed dual authentication structure. OBU authentication is conceptually simple: OBU manufacturers vouch for OBUs. The RSU authentication structure is the classic hierarchical PKI which maps to the existing political authority structure. National *Certificate Authorities* (CAs) vouch for Regional CAs which vouch for Local CAs, etc. Authentication of OBUs will be done via a privacy preserving mechanism that simply vouches for the fact that the unit is a valid OBU. Authentication of RSUs and public safety OBUs will be done via traditional public key and certificate mechanisms which have been tuned to work well in the constrained VSC environment. Each level of certification is increasingly restrictive. Thus, Regional CAs can only issue certificates for their own regions, local CAs can only issue certificates for their own localities, etc. This limits the amount of damage that compromise of a single part of the authentication hierarchy can do. We describe an alternate structure for RSU authentication in Section 5.3.1.

4.3.2 Communications Architecture

Like other communication systems, VSC is structured as a series of layers, as shown in Figure 8. At the top of the stack is the Safety Application Layer, which is responsible for handling safety-related information. Next is the Security Layer, which is responsible for securing the safety messages. The bottom three layers, DSRC Messaging, 802.11 MAC (Media Access Control), and 802.11 PHY (Physical layer), take care of the actual network data transmission. Messages which are being transmitted move down the stack from the Safety Application Layer to the airwaves. Messages being received move up the stack from the airwaves to the Safety Application Layer.

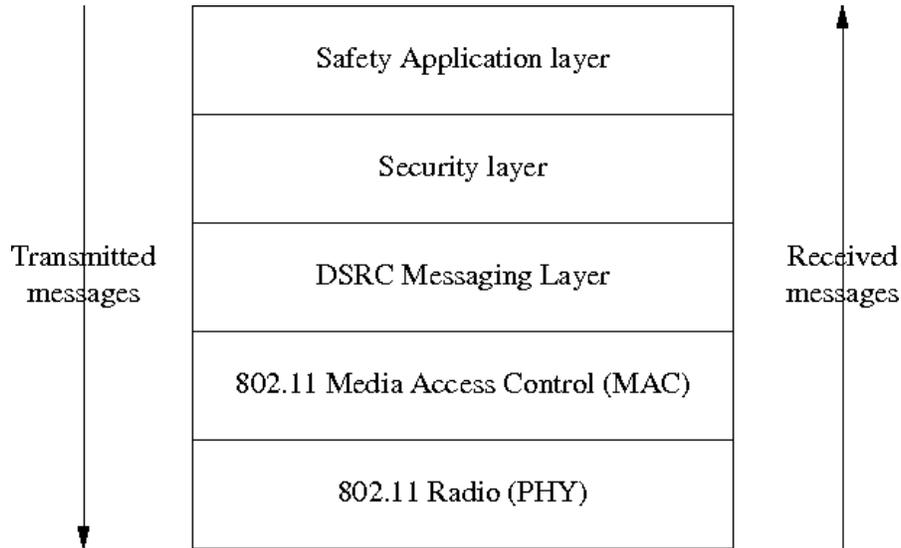


Figure 8: VSC Communications Architecture

4.3.3 Message Authentication

Once OBUs and RSUs have their keying material, they can then transmit messages using DSRC. When a unit wants to transmit a message, it first appends a timestamp and position indicator. These are derived from a trusted source and therefore must be appended by the security layer. It then applies a digital signature using its keying material and attaches the signature to the message along with some kind of key identifier (what kind of identifier depends on the architectural choices described below, but it is obviously some kind of bit string) and broadcasts the message to the receiver using DSRC. Note that we are omitting details like version numbers and algorithm and type identifiers in this presentation.

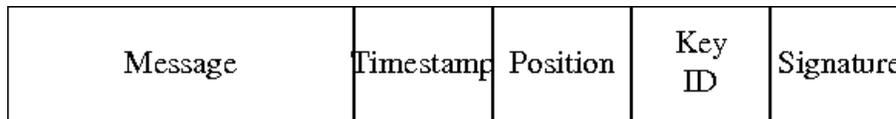


Figure 9: A protected Message

When a unit receives a message, it uses the key identifier to find the appropriate key to use for verification. The key identifier is typically either a certificate or a key index. If the former, the unit will verify the certificate. If the latter, the unit must already have the key in its cache – we

⁴ Note that there has been some discussion of introducing an intermediate channel selection layer in between the 802.11 MAC and the Security Layer. The Security Layer does not provide any integrity protection for this layer.

assume for the moment that it is pre-loaded. (The following sections provide more detail on this point.) Once the key is identified, the unit uses it to verify the signature on the message using the appropriate digital signature algorithm (again, this is covered in more detail later). All of this processing happens in the Security Layer. If the signature is invalid, the message is rejected. If the signature is valid, the receiver verifies that the key used to sign it has the appropriate permissions for the kind of message received. For instance, OBUs are not allowed to send Curve Speed Warning messages. This check may also entail verifying that this keying material is valid for this location. For instance, California RSUs should not be transmitting safety information in Denver. If any of these checks fail, the message is rejected. If they all succeed, the message is accepted for processing.

4.3.4 Management, Revocation, and Updates

Any security system needs periodic updates. These may include:

- *Updated/patched code.* Even the best software inevitably has bugs. The traditional way to fix these bugs is to release new versions of the software with the bugs fixed. Uptake of such patches is slow even with general purpose computers [11] and would most likely be minimal with vehicles if driver intervention is required – people just do not think about having to upgrade their cars. Therefore, a new approach will be needed.
- *Propagation of revocation lists.* It may be necessary to publish lists of revoked (known to be compromised) units.
- *Keying material refresh.* In order to minimize the scope of compromise and the size of revocation lists, it is desirable for units to periodically obtain fresh keying material.

In order to fulfill these needs, it is necessary to have a mechanism for propagating relatively large chunks of data to users. This data would be authenticated via the same sort of mechanisms as ordinary messages, but would most likely not be structured as individual messages but rather as large files split over multiple messages in some fashion. These files will themselves be signed, using, for instance Cryptographic Message Syntax [13].

4.3.5 Fail-Open versus Fail-Closed

Because our design requires that units be periodically updated, the question arises of whether units should continue to function when they are unable to receive updates. For instance, if a unit receives a message from a unit without having a current certificate revocation list, it cannot be sure that the message is from a valid unit and must therefore decide between accepting a potentially invalid message (failing closed) and rejecting a potentially important safety message (failing open).

In this case, how to behave is an implementation decision, although the VSC protocol could require a fail-open or fail-closed behavior as a matter of policy. However, some designs inherently fail-open. For instance, if units need to periodically obtain new keys (e.g., as in the design described in Section 5.5.4.5), then they will be unable to transmit authenticated messages if they do not obtain new keying material, which could be more difficult in low-population density areas (depending on the flooding mechanism and available infrastructure). We believe that in general the system should adopt a fail-open policy. When there is a substantial question as to the validity of a message it should be rejected.

4.4 RSUs and Public Safety OBUs

RSUs and public safety OBUs have no particular anonymity requirement placed on them by the threat model and system constraint assumptions, and can therefore use a straightforward *Public*

Key Infrastructure (PKI)-style solution. PKI is not the only approach that would conceivably work, however it is a standard and well-understood security design, supporting systems as diverse as the Web e-commerce and the Department of Defense communications system. Using standard tools such as PKI allows us to avoid the risks attendant with any new security invention.

In the RSU/PSOBU PKI system, each unit has an asymmetric key pair. The public key is signed by the local CA, and the units use their private key to sign all of their messages. Certificates contain restrictions indicating what sort of messages may be signed using the corresponding key.

Note that upon initial examination it appears attractive to have RSUs do without keying material entirely and merely have them rebroadcast canned messages, e.g., "There is a stop sign at the following geographic coordinates". However, a number of RSU types (such as traffic signals) and nearly all public-safety OBUs are dynamic; they send different messages reflecting their current state. In order to support dynamic RSUs while still avoiding replay attacks, it is therefore necessary for the RSUs to have their own keying material and generate signed messages.

4.4.1 Authentication Structure

Conceptually, authority follows geographic lines of control and as one descends the certificate hierarchy, the scope of control for any given CA becomes smaller. Thus, for instance, Philadelphia would need Pennsylvania's approval (tacit or otherwise) in order to issue certificates to RSUs. This authority structure may or may not be reified in the certificate structure. In the Section 4, we considered two architectures, which we present here.

The topology of the PKI impacts on efficiency and ease of deployment. It also has political ramifications. The PKI is primarily responsible for the issuance of digital certificates and revocation information. Certificate Revocation Lists (CRLs) are the most common form of revocation information. Imagine we have an RSU which has the following place in the hierarchy:

Country	US
State	Virginia
County	Fairfax
RSU #	10048

We need to consider two major alternative architectures *hierarchical* and *flat*.

Hierarchical PKI

The PKI topology mirrors the levels in the naming hierarchy. For example, the US DOT operates a CA for the top level, issuing certificates to each of the States. Then, the States operate CAs, and they issue certificates to each of the counties and cities within the state borders. Finally, the local governments operate CAs, and they issue certificates to the RSUs.

This alternative directly matches the expectations of most people familiar with governmental structures. A single trust anchor is needed to represent each country. A superior international organization could be created to serve as the top-level CA, but it has many political issues, and it imposes additional efficiency concerns.

This hierarchal structure imposes a burden on the OBU. Using the above example, four signature validations are needed to verify a message from RSU 10048. They are:

1. Validate the signature on the VA certificate with the US DOT public key.

2. Validate the signature on the Fairfax County certificate with the VA public key.
3. Validate the signature on the RSU certificate with the Fairfax County public key.
4. Validate the signature on the message with the RSU public key.

Further, revocation information provided by each of the CA ought to be checked. If CRLs are employed, three additional signature validations are needed. Each of the CRLs ought to be small since each CA is not responsible for a large number of certificates.

Deployment requires each of the superior organizations to be operational prior to a subordinate governmental entity. Therefore, Fairfax County cannot install their first RSU until the US DOT and the Commonwealth of Virginia are both fully operational.

Flat PKI

A single CA issues all of the certificates within a nation, but the naming hierarchy is unchanged. For example, the US DOT operates the CA for the top level, but it accepts requests for certificates and revocations from authorized registration authorities (RAs) in each of the states, counties and cities. The process for authorizing RAs will likely follow the natural governmental structure, but once authorized, the RA can communicate directly with the CA.

All of the political concerns are transferred to the RA registration process. And, the CA is responsible for ensuring that one local government cannot request certificates that are affiliated with other jurisdictions.

This flat structure reduces the burden on the OBU. Using the above example, two signature validations are needed to verify a message from RSU 10048. They are:

1. Validate the signature on the RSU certificate with the CA public key.
2. Validate the signature on the message with the RSU public key.

Revocation checking can be problematic in this structure, since all of the revoked certificates are listed in a single CRL. While only a single signature is needed to validate the CRL, the number of entries on that CRL can become quite large. A large CRL takes significant bandwidth to download. This concern might be mitigated by performing the download at gas stations where the vehicle is stationary.

Deployment only requires the national CA to be operational. Therefore, Fairfax County cannot install their first RSU before the Commonwealth of Virginia has installed any capabilities, assuming that the registration process can be accomplished by paperwork.

This alternative has an additional concern. The national CA must be highly available. In the first alternative, the outage of any single CA has very little impact on the overall system, but in this alternative, an outage in the national CA can have major ramifications. Therefore, an architecture that includes concurrent operations from multiple physical locations is needed, making the national CA more expensive in this alternative.

Hybrid Alternatives

Hierarchical and flat PKI topologies represent the extremes of a continuum. Two mechanisms exemplify other points along the continuum: partitioned CRLs and indirect CRLs. Partitioned CRLs are used to ensure that CRLs never get too big. The population of certificates is divided among a group of small CRLs, and the certificate contains a pointer to the CRL that needs to be checked for revocation. For example, the first 1,000 certificates issued point to one CRL, the second 1,000 certificates point to the next CRL,

and so on. This ensures that none of the CRLs is ever larger than 1,000 entries. Of course, most should be much smaller than the maximum.

A better method of portioning the CRLs for the DSRC application is geographically. This allows an OBU to obtain and validate the CRL for a specific area as it enters it. One approach might be to partition the CRLs by ZIP code. Another approach would be to use the grid defined for the World Aeronautical Chart. Indirect CRLs allow one CA to issue the certificate and another to issue the associated CRL. In this way, a national CA could issue each certificate, but the certificate would contain a pointer to the locally-administered CRL that needs to be checked for revocation. Certification path checking is a bit more expensive than in a hierarchical PKI, but revocation is handled locally.

4.4.2 Certificate Format

Because of the packet size limitations explained in Section 4.2.6, conventional X.509 certificates [12] are unsuitable for use with VSC Security. They are excessively large and there is a large semantic gap between X.509 distinguished names and the names which make sense here. Although it is in principle possible to shoehorn VSC information into X.509 certificates, the result would be fairly unwieldy. In addition, because interoperability with conventional PKIs is not a priority and we do not anticipate using the signature algorithms that are commonly used with X.509 certificates, there is no significant advantage to using X.509 rather than a custom format.

In this section, we sketch the outlines of a certificate format for VSC Security but stop short of providing a bits-on-the-wire description. VSC certificates consist of at least four data items:

- The public key of the certificate holder.
- The scope of the certificate.
- The validity window of the certificate (expiry time).
- A signature over the certificate.

The public key of the certificate holder and the signature are straightforward cryptographic data. The validity window can be easily represented in one of a large number of more or less equivalent formats (UTC time, seconds since the epoch, etc.) However, the scope requires more discussion.

Scope

In RSU/PSOBUs certificates all important authorization information in a certificate must be carried in the scope field. We are concerned with two kinds of authorization information:

- Geographic – the physical regions during which this certificate may be used.
- Functional – the types of assertions that the holder of this certificate can make.

We discuss each of these kinds of scope in turn.

Geographic Scope

The purpose of geographic scope is to restrict certificates to the geographic regions for which their operators have authority. For instance, the California Transportation Agency would find it undesirable for the Illinois Transportation Agency to be able to install RSUs that broadcast information about California traffic signals, since this would mean that compromise of an RSU in Illinois would open up attacks in California. In order to

prevent this, each certificate must contain an indication of which geographic areas the certificate may be used in.

We propose that these restrictions be encoded in one of three ways:

- A radially symmetrical zone
- A polygon
- A sequence of rectangles

The specification for a radially symmetrical zone is a single lat/long coordinate specifying the center and a radius of the zone. The intention is that this mechanism is used for immovable objects like RSUs.

A polygonal restriction is a sequence of geographic locations (expressed as lat/long coordinates), which are then connected by straight lines in a connect-the-dots fashion. These points and lines define an enclosed region (or, perhaps, regions), which delineates the scope of the certificate. The certificate may speak for any point in that region. The intention is that this sort of scope will be used for CAs and PSOBUs.

A sequence of rectangles represents a middle ground between a circle and a polygon. It allows a more complex region to be defined than does a circle, but allows less complexity than a polygon and is therefore easier for a receiver to evaluate.

Note that there is some redundancy between the different representation formats. In particular, polygons can be used to emulate the other formats to a great degree. At the end of the day, experience may show that it is advisable to decide on a single format. However, initially we consider it wise to offer some variety for experiment.

In principle, this technique can be used to define the validity region to an arbitrarily fine granularity. However, in practice the need to keep message sizes down requires that the number of points be kept relatively small. This inevitably creates some areas where the certificate restrictions do not precisely match geographic boundaries. In practice, we do not anticipate this being a problem since bordering regions in general need to be able to coordinate their activities. This sort of restriction is intended primarily to proactively contain compromise between distant geographic regions. One way to produce finer grained boundaries while reducing message size is to issue multiple certificates, which together cover the entire permitted zone.

Functional Scope

It is extremely desirable to restrict the uses to which a certificate can be put. For instance, rural traffic signals are often isolated and poorly monitored. If an attacker could compromise such a signal he or she should not be able to then use it to impersonate a police car. In order to prevent this kind of attack, certificates must contain functional restrictions.

As with geographic scope, functional scope should be thought of as a tree, with restrictions becoming increasingly tight the further one moves away from the root. A proposed tree is shown in Figure 10.

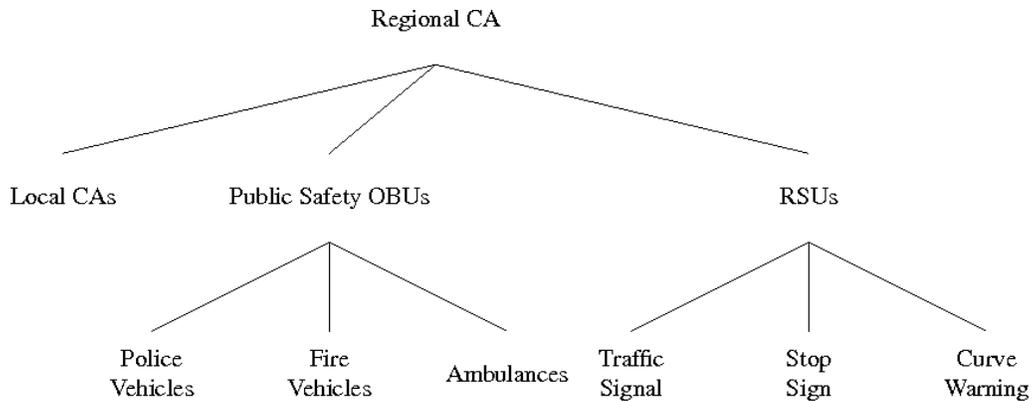


Figure 10: Functional Scope for RSUs and Public Safety OBUs

Note that it is possible – indeed likely – that a CA will sign other CAs as well as RSUs/OBUs. For instance, the California State CA might sign certificates for both the OBUs for the California Highway Patrol and for the county CAs.

No Identities

Note that unlike conventional certificates, DSRC certificates do not contain an identity string. All of the important authorization information is carried in the scope field and therefore we can save space by not including an identity payload at all.

4.4.3 Registration and Certificate Issuance

Any certificate system needs some way for units to get certificates. In VSC, these mechanisms fall into two categories: initial registration and certificate issuance.

Initial Registration

When a new PSOBU or RSU first enters the system, it is un-initialized and must be introduced to its CA. The introduction process creates a binding between some shared cryptographic credential and an entry in the CA database. For VSC, this is done one of two ways: either via direct connection or remotely. In either case, the first step is for the CA administrator to determine which permissions the unit should be allocated. This is done using non-technical mechanisms which are not specified by VSC. For instance, when initializing traffic signals, the CA administrator may be the person installing them or may communicate directly with the installer.

In the direct connection case, the unit is fitted with an interface (e.g., USB or Firewire) and the administrator physically plugs the unit into the CA. The administrator then interrogates the unit for its public key and records that key in its database, bound to the unit's identity and permissions.

In the remote case, the unit's public key (or more likely a digest of the public key) would be delivered along with the unit (for instance, embossed on the outside like a serial number). The operator of the unit would then tell the CA administrator the public key (or digest) and he would enter it into the CA database. Note that this process could also be performed via Web or some other remote access mechanism, provided that the unit operator authenticates to the CA.

Certificate Issuance

Once the initial registration has occurred, the CA can then issue certificates for the unit. The certificate permissions are controlled by whatever settings are in the CA's database. The simplest way for certificate issuance to operate is for the CA to periodically issue appropriate certificates for the public key stored in the database. Alternately, the CA might require the unit to provide a new key but authenticate the request using the initially registered key.

For RSUs, we anticipate that transit agency personnel would periodically visit the unit to install a new certificate. However, RSUs could potentially be refreshed over the Internet if network service is available. PSOBUs will likely have their certificates refreshed at their "home bases". For instance, when the vehicle is driven into the garage, it could establish a wireless connection with the local CA and get a new certificate.

4.4.4 Message Authentication

Once the keying material is established, RSU/PSOBU message authentication is accomplished by traditional digital signature techniques, in the flavor of CMS [13]. Each message has a digital signature attesting to its authenticity and an identity certificate attesting to the identity and permissions of the sending unit. The receiving unit can then verify the message and act appropriately.

Message Format

The PSOBU/RSU message format contains the following items:

- Payload type identifier
- VSC safety application information (payload)
- Locally-unique message identifier
- Timestamp
- Position
- Sender's certificate
- Signature over entire message

Note that the security system is completely blind to the contents of the safety payload. It only knows the type (for de-multiplexing on message receive). The Security Layer is simply responsible for verifying that the message is authentic. The VSC Safety Application Layer must then determine if the message is appropriate, as described in Section 5.4.4.3. Figure 11 shows such a message. The shaded sections are signed.

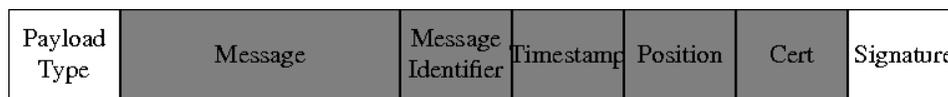


Figure 11: A protected message

Signature

When a safety message is presented to the Security Layer for processing, it must perform the following stages:

1. Generate a unique message identifier. The intent of this message identifier is to uniquely identify this message among others sent by the same unit at the same time (in cases of insufficient time granularity). This identifier can be a simple short counter or can be generated by a random number generator [14].
2. Form a temporary message consisting of the message id, the payload, and the sender's certificate.
3. Sign the temporary message.
4. Append the message signature to the temporary message to form the final message to be transmitted.
5. Pass the final message down to the DSRC Messaging Layer.

This is a fairly conventional message processing procedure, roughly analogous to that used in IPsec ESP [15].

Verification

When a unit receives a PSOBU/RSU message, it must first verify the authenticity of the message. This requires two sets of checks, one performed at the Security Layer, one performed at the Safety Application Layer. The Security Layer's job is to verify that the message is *cryptographically correct*. This means that it was correctly signed by a valid user. Verifying this requires verifying both the signature and the certification path.

Each message contains only the certificate of the sender. In order to validate that certificate, the receiver must construct a path of certificates up to a trust anchor (a certificate which is trusted because it is built into the OBU). At each stage in that path, the recipient must verify that:

- The signature on the certificate is valid.
- The restrictions on the certificate allow the current use. For instance, certificates for CAs (often called intermediate certificates) must be tagged as CAs and have the correct geographic scope for the certificate they are signing. Note that this means that the regional scope for any certificate must be entirely within the regional scope of its parent.
- The certificate is current and not expired.
- The certificate has not been revoked.

Note that units are expected to cache verified certification paths, thus reducing the burden of re-verification for new certificates. For instance, once a unit has verified the Palo Alto Police certificate, it can verify any individual police car by using the cached path, and then checking the car's individual certificate. Once the certificate has been verified, the receiving unit must then verify the message. This is a simple matter of verifying the digital signature using the public key in the certificate. The fact that a signature is valid may be cached indefinitely, however the validity of the certificates must be checked whenever a new CRL is received or the current one expires.

In order to prevent replay attacks, each message contains a timestamp which is generated inside the trust boundary of the transmitting unit. The Security Layer must maintain a list of all recently received messages (a window of 1-5 seconds is probably appropriate) and reject messages that either (1) bear timestamps outside that window or (2) have been replayed. The Safety Application Layer must also check that the position described in the message is within the geographic scope of the sender's certificate.

Once the Security Layer has verified the authenticity of the message, it passes it on to the Safety Application Layer for processing. The Safety Application Layer must then verify that the certificate that was used can be used to sign a message of this type. For instance, a Stop Sign RSU might sign a message claiming to be a Curve Speed Warning. This must be detected at the Safety Application Layer because the Security Layer does not have any visibility into the Safety messages. The Safety Layer must also verify that the message is geographically and temporally relevant (different applications have different relevance requirements). Only if all of these checks succeed is the message accepted for processing.

The intent of placing this processing in the Security Layer is to allow new applications to be deployed with their own validity requirements. An alternative design would be to have a Security Layer which accepted configurable policies for message validity. The Safety Layer could then describe which applications were to receive which policies and the Security Layer could enforce the correct policy based on the payload type field.

4.4.5 Propagation of Certificates

In order to verify a message, it is necessary to have the entire certification path up to a trust anchor. However, in general the entire path will be too large to fit into time critical messages. Therefore, there must be some mechanism to pre-load verifiers with these certificates. Because certificates are geographically restricted, there is a straightforward mechanism to arrange for certificate pre-loading. RSUs near geographic boundaries periodically broadcast the entire certification path for their region. The RSU could be Internet connected and periodically contact the CA for refreshed intermediate certificates or CRLs.

Thus, when vehicles enter a new region, they automatically acquire the new certificates and cache them. Because these messages are not time critical, they can be fairly large, up to the maximum packet size of the network. In the event that the maximum size is too small, the certificates may be spanned across multiple network packets using the technique described in Section 5.5.2.2.

4.4.6 Identifying Compromised Units

Identifying compromised RSUs and PSOBUs is relatively straightforward. We need to consider three basic cases. First, we may know the real-world identity of a compromised unit, e.g., a serial number. It is expected that CAs will keep records of which units had which key pairs, and therefore it is straightforward to revoke that particular unit and to ensure that no future certificates are issued to that unit.

In the second case, we may have captured some set of messages that are considered bad. Because certificates signed by PSOBUs and RSUs contain the certificate of the signing unit, we immediately can find the random identifier. This identifier can then be used as the key for a CA database lookup to find the real world identity of the compromised unit. That unit's certificate can then be revoked and/or the unit can not be issued any future certificates.

Finally, consider the case where an attacker compromises a unit and anonymously publishes its private key. Although the user cannot be identified, we can still identify the compromised unit in two ways. First, the private key can be used to generate the public key, which can then be looked up in CA databases. Second, the private key must be used with the associated certificate. Once the certificate is found, compromise can then be handled as in the previous case.

4.4.7 Certificate Revocation

In any PKI system, there are two basic mechanisms for dealing with compromised keying material: revocation lists and timeouts.⁵ Choosing which mechanism to use is a tradeoff between the cost of propagating certificate revocation lists and the cost of refreshing keying material. Where the best tradeoff is depends on the extent of compromise – and therefore of revocation – and the required level of timeliness. Because we cannot predict *a priori* the extent of compromise, the best option is to design a system that incorporates both methods of compromise containment. When a certificate expires, it can generally be refreshed using the certificate issuance mechanisms described in Section 5.3.3. However, revocation requires a new mechanism.

Revocation lists are easy to create. They are simply a signed list of all of the non-expired keys which have been revoked. However, the lists must then be distributed. Because, like certificates, they are geographically bounded, they should be distributed at the borders of geographic regions. In the event that they are too large to fit into a single packet, they may be spanned across multiple packets, as with certificate distribution.

4.4.8 Choice of Algorithms

Although this is a traditional asymmetric key/digital signature system, a careful choice of signature algorithm nevertheless is required due to message size and performance considerations. Asymmetric signature systems differ along four dimensions that concern us:

- Signature size (S_{sig})
- Public key size (S_{pub})
- Signing time (T_{sig})
- Verification time (T_{vfy})

The first two parameters control the size of the signed messages. The second two parameters control the performance of the system, and thus directly impact latency.

Message Size

Any signed message must include:

- The certificate (size $S_{sig} + S_{pub} + S_{rest}$) where S_{rest} is the size of the identities, validity period, and restrictions and varies depending on the granularity of the geographic access controls but is constant across signature algorithms.
- The signature over the message (size S_{sig})

Thus, the minimum size for a given signed message (with an empty payload) using a conventional signature scheme is $2 \times S_{sig} + S_{pub} + S_{rest}$.

The magnitude of S_{rest} depends on the granularity of the scope restrictions in the certificate. At minimum, S_{rest} will contain a 32 bit time field. The scope field consists of a series of points defining the shape of the allowed geographic region. Each point will be approximately 8 bytes in length. Thus, a plausible value for S_{rest} is 32 bytes (four points). A radially symmetrical zone will be smaller: 8 bytes for a point and probably no more than 2 bytes for a radius. Note that S_{rest} is independent of the choice of cryptographic

⁵ In principle, one could also use an online status protocol such as OCSP [16], but in practice, such mechanisms are impractical in a broadcast situation.

algorithm. In the rest of this section, we will be discussing only the size of the cryptographic portions. An additional S_{rest} bytes of overhead will always be required.

Performance

In order to send a message, the sender must perform a single digital signature, costing T_{sig} . In order to verify a message, the recipient must at a minimum verify the message signature (time T_{vfy}) and will likely have to verify at least the certificate signature (time T_{vfy}). If we assume that the rest of the signatures are cached, the total verification time is $2 \times T_{vfy}$. Thus, the total latency introduced by security is $T_{sig} + 2 \times T_{vfy}$.

Message sending throughput depends entirely on signing time. Thus, a sending unit can transmit $1/T_{sig}$ messages per second. Message receiving throughput depends on verification time. Thus, a receiving unit can receive between $1/T_{vfy}$ and $1/2 \times T_{vfy}$ messages per second, depending on how many of the messages are from senders whose certificates have previously been verified.

The actual performance of any algorithm depends on the processor it is running on. For reference, we quote numbers from the MIRACL library of Mike Scott running on a 450 MHz Pentium III [17]. There's nothing special about this library, except that Scott provides measurements for a large number of different algorithms, thus enabling easy performance comparisons between algorithms.

Patent Status

Finally, we must consider the patent status of each algorithm. If OEMs must license patents in order to deploy the VSC security system, then this increases their costs. We provide a description of what we know about the patent status of the algorithms we describe. This information cannot, however, replace a thorough legal search.

RSA (Rivest, Shamir, Adleman) Algorithm

RSA [18] is the world's most widely used signature algorithm. At the standard 1024 bit strength, it is fairly fast for signature (17 ms) and extremely fast for verification (.11 ms, public exponent=3). It would meet our performance targets with minimal hardware acceleration.

The RSA verification step is extremely fast since it uses small exponents, so that even a fairly small ASIC core can easily meet the latency and throughput goals for verification. Fortunately, the RSA signature goal (i.e., 5 msec latency) can also be achieved reasonably easily. For example, a new Hifn chip uses slightly less than 100K gates in a 0.13 micron CMOS process to achieve 388 RSA 1024-bit signatures/sec, running at 200 MHz. This is a latency of under 2.6 msec. The same core can perform RSA 1024-bit verifications, using a 17-bit exponent (65537), with a latency of under 0.2 msec. An area of 100K gates, when added to an existing chip, currently results in an incremental silicon cost of under \$0.50 in volume. If this design were to be fabricated as a stand-alone chip, the packaging and testing costs would probably raise the per-chip cost to \$2 – \$4, in large volume. The power consumption of such a core, at full duty cycle, would be (very roughly) on the order of 0.25 Watts.

However, the RSA signature size is unworkably large. At 1024 bits, $S_{sig} = S_{pub} = 128\text{bytes}$.⁶ Thus, the minimal cryptographic overhead for a message is 384 bytes. It is

⁶ This assumes that everyone uses the same public exponent so there is no to add it to the public key. If the exponent varies, add a few bytes overhead.

possible to aggregate multiple RSA signatures into a single signature[19] (and this requires using a non-standard form of signature) thus reducing the minimal overhead is 256 bytes, which is still likely to be too large.

There are no intellectual property costs associated with using RSA, because the patent on RSA [20] has expired.

DSA (Digital Signature Algorithm)

The other commonly used signature algorithm is the *Digital Signature Algorithm* (DSA), defined in Federal Information Processing Standard (FIPS) 186-2[21]. DSA is roughly twice as fast as RSA for signature (8.8 ms) but is far slower for verification (10.75 ms). Therefore, significant hardware acceleration would be required in order to obtain acceptable performance.

DSA signatures are 40 bytes long, but the DSA public key is as large as RSA. If a common parameter group is used, at a 1024 bit strength, $S_{pub} = 128$ bytes. Thus, the total overhead is 208 bytes, less than RSA even with aggregation. However, since our target is 100 bytes, this is still larger than desirable.

The major computational step in a 1024-bit DSA signature is a single modular exponentiation, with a 1024-bit modulus and a 160-bit exponent. To first order, all other operations involved in the signature are only minor additions to this time. Similarly, the dominating computation cost for a 1024-bit DSA verification consists of two such exponentiations. A third-party vendor (Athena Group) currently offers a modular math core that can conservatively achieve the following latencies, running at 200MHz:

1024-bit DSA signature	0.25 msec
1024-bit DSA verify	0.50 msec

The area is about 250K gates. These numbers are all approximate and may end up slightly better or worse, depending on the silicon process and the time taken in synthesis and physical design. In any case, these performance numbers meet the desired goals, and the incremental silicon cost of such a core would be in the \$1-2 range. As a stand-alone chip, the volume pricing would probably be in the \$3-\$6 range. The power consumption would be very roughly on the order of 0.5 W.

We know of two U.S. patents that are asserted to cover DSA. U.S. Patent 5,231,668[22] and U.S. Patent 4,995,082[23]. The '668 Patent is assigned to the United States of America and is available royalty-free. The '082 patent is not royalty-free. There has been debate about whether or not the '082 patent is required for implementations of DSA and many vendors in the US have not obtained patent licenses. A thorough analysis of this patent should be performed.

ECDSA (Elliptic Curve DSA)

As described previously, the primary problem with DSA is that the public key is too large. *Elliptic Curve DSA* (ECDSA) is effectively DSA over an elliptic curve. Elliptic curve algorithms such as ECDSA are as strong as integer algorithms such as RSA or DSA with smaller key sizes. In particular, ECDSA is roughly as fast as DSA, but at the 80-bit security level (equivalent to a 1024-bit RSA key) has only a 160-bit (20byte) key size. The signature is the same size as DSA. Thus, the minimum space overhead with ECDSA as the signature algorithm is $40 + 40 + 20 = 100$ bytes. It may also be desirable to use keys stronger than 160 bits for high-level CAs, thus providing increased resistance to analytic attack.

Elliptic Curve algorithms are not as widely implemented in silicon as RSA and DSA. However, we are able to make some estimates as to performance and gate costs. We believe that we can achieve the target performance of 2000 ECDSA verifications per second using approximately 75,000 gates. This should add a cost of about \$0.30 / VSC unit and consume less than 0.2 watts. See Appendix A for the reasoning behind these estimates.

Certicom has asserted a number of patents on efficient implementations of elliptic curve cryptosystems [24]. The validity of these claims themselves should be investigated and appropriate patent licensing (as necessary) pursued.

ECDSA can also be used in an "implicit certificate" mode where the public key and the signature are combined, thus reducing the space required to represent the public key. [25] This technique may be patented by Certicom. Again, we advise the pursuit of patent licensing as appropriate.

BLS (Boneh, Lynn, Shacham) Algorithm

Boneh, Lynn, and Shacham [26] describe a technique for using the Weil pairing to produce a very short signature (approximately 20 bytes). The public keys are of equivalent size. In addition, BLS signatures can be aggregated, so the minimal space overhead for security could be as low as 40 bytes. Unfortunately, BLS signature verification requires two pairings and is therefore extremely slow. Accurate numbers are not available, but our estimate is on the order of 40 ms (though two aggregated signatures can be verified at once, so this is the total overhead to verify one message). Very extensive hardware acceleration would be required to make the performance of BLS acceptable.

Algorithm Analysis Summary

We consider the best algorithm choice to be ECDSA. RSA and DSA have too large a space overhead to be practical in this application. Because BLS is very new and has only seen modest amounts of review, we do not believe it is appropriate to standardize on at this time. Thus, we recommend that ECDSA be chosen as the signature algorithm for RSUs and PSOBUs.

4.5 Authentication for OBU Messages

The situation with end user OBUs is more difficult than that with RSUs. Although the authorization problem is in principle much simpler (the OBU merely needs to establish that it is a valid OBU, rather than one with any particular set of characteristics), it is rendered much more difficult by the need to preserve privacy. In the rest of this section, we discuss the relevant criteria for deciding on a scheme and then describe a number of potential schemes for OBU authentication.

4.5.1 Selection Criteria

When deciding which approach to use for OBU message authentication, we need to consider four issues: privacy, performance (size and bandwidth), management complexity, and resistance to compromise.

Privacy

Beyond the desire for sender authentication, it is assumed in the threat model and system constraints that end user OBUs have an increased need for privacy. In this context, we are primarily concerned with two security properties:

1. *Anonymity* – it is not possible to determine a vehicle's identity from its transmissions.
2. *Unlinkability* – it is not possible to determine that multiple transmissions were from the same source.

Note that unlinkability is a stronger security property than anonymity, since a lack of anonymity automatically implies linkability.

When considering these properties, we have to be concerned with two kinds of attackers: ordinary users and insiders (Classes 1-3 versus Class 4 in the terminology of our threat model). For instance, it might be possible for OBU vendors to determine users' identities even if it is not possible for ordinary users to do so.

Performance

Authentication for end user OBUs has similar performance issues to those for RSUs and PSOBUs: time critical messages need to have small authentication tags, and both signature and verification must be quick.

Containing Compromise

As discussed in Section 5.3.6, identifying and containing compromised RSUs and PSOBUs is straightforward. Any bad message can be captured and used to identify and revoke the compromised unit. However, the requirement for privacy for end user OBUs makes containing compromise of such units more difficult. There are three important scenarios for detecting compromise and deriving a discriminator for the compromised unit.

1. The investigator captures one or more messages using the compromised unit's keying material.
2. The investigator discovers the private key of the compromised unit. This would be likely to occur if an attacker posted their private key on the Internet or if the investigator obtained a cloned unit.
3. The investigator obtains the identity (e.g., serial number) of the compromised unit, perhaps by ordinary investigative work.

Some of the schemes we will be considering allow compromise to be contained only in a subset of these scenarios.

Note that we are primarily concerned here with containing compromise by Class 3 attackers (those who have successfully broken their OBU). Class 2 attackers (those with un-tampered units) can be contained fairly simply by including a message identifier in all messages and incorporating an algorithm in each OBU that will shut down the unit on command. When a bad message is detected, an authority can broadcast via RSUs a shutdown message that says "whatever unit generated this message, shut down." One possible attack on such a shutdown command is to shield one's compromised OBU except when it is being used for live attacks. This attack can be countered by tying the shutdown command to system updates so that units must process it in order to remain active in the system.

Management Complexity

Finally, we need to consider operational issues. The number of end user OBUs will be very large and, unlike RSUs and PSOBUs, they are not directly managed by the authorizing agency, which in this case is their manufacturer under the proposed security

architecture model. Thus, management of the units becomes a serious issue. For instance, if a new software revision needs to be rolled out, some mechanism is needed for arranging that all the affected units get a copy. Additionally, when units are compromised, it may be necessary to publish revocation information. Unlike PSOBUs and RSUs, revocation information for OBUs is not geographically limited and, therefore, must propagate to all OBUs. Therein, this distribution problem is more difficult.

4.5.2 Building Blocks

The schemes described in the remainder of Section 5 of this document make use of a number of more or less standard primitives. We describe them here briefly for readers who may not be familiar with them.

Broadcast Encryption

In some of the schemes that follow, a number of OBUs share the same keying material. That keying material must be periodically refreshed, which requires the ability to send them encrypted refresh messages. This is isomorphic to a common problem encountered in the security literature: how to encrypt a single message to a large group of people with minimal overhead. The typical setting is that you have a large group of receivers of size N . You want to encrypt a message to all of these receivers with the exception of a subset of size r who are excluded. The naive way to do this is to have each receiver have a single key and separately encrypt to each receiver. This produces a very large message (linear in size $N-r$).

A number of superior solutions have been proposed. The current state of the art is two schemes described by Naor et al.[27]. We describe their simpler Complete Subtree scheme here.

We have a universe of N receivers, numbered from 1 to N , as in U_1, U_2, \dots, U_N . Imagine each user as the leaves of a binary tree, thus U_1 and U_2 are under a common interior node, as are U_3 and U_4 , and so on. A small example of such a tree with $N = 4$ is shown below.

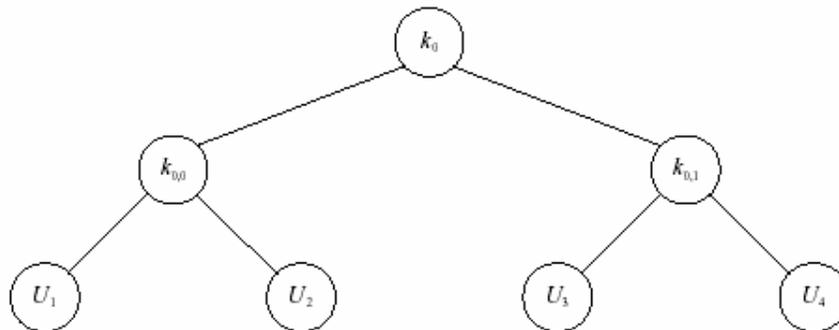


Figure 12: A broadcast encryption tree

Each node in the tree is assigned a symmetric encryption key. Each leaf (a.k.a. “unit”) is given its own key plus all the keys on its path to the root. Thus, for instance, U_1 has its own key, plus $K_{0,0}$ and K_0 . In order to encrypt a message, the sender encrypts under a set of keys whose sub-trees cover the entire unrevoked set.

For instance, if no units have been excluded, the sender simply encrypts under K_0 , which is known to all units. Now consider what happens if U_4 has been revoked. In that case, the sender encrypts under both $K_{0,0}$ (known to U_1 and U_2) and under U_3 's key.

The efficiency of this scheme declines when a large number of units are excluded. In general, the number of separate encryptions that the sender must perform (and hence the size of the ciphertext) is $r \log(N/r)$ blocks. However, it is never any worse than the naive scheme. This scheme requires $\log N$ keys worth of storage at each receiver.

Naor et al. also describe a superior scheme called "Subset Difference". Subset difference is quite complicated and so we defer the discussion to Appendix C and simply state its properties. Subset Difference has a superior message size of $2r - 1$ blocks but the tradeoff is increased storage at each receiver of $1/2 \log^2 N$ keys. In general, message size is more critical than storage, which is relatively cheap, so the VSC team suggests encrypting using Subset Difference.

Erasure Codes

Another common problem in broadcast/multicast networking is the need to send large chunks of data in an environment where packets may be lost. In point-to-point networking, this is handled with reliable data transfer protocols such as TCP [28], but the acknowledgements required by such protocols are inconvenient in a broadcast environment. A superior approach in these environments is to use *erasure codes* [29].

The idea behind an erasure code is straightforward. Say there is a message M , which is p packets long. Instead of simply breaking it up and broadcasting it in p packets, we encode it into a group of p' packets $M'_1, M'_2, \dots, M'_{p'}$. The encoding is done in such a way that *any* p -sized subset of the new packets is sufficient to recover M . The sender then broadcasts all p' prime packets. Any receiver that receives an appropriately sized subset has received the message.

In situations where the loss rate is known, one can simply choose an appropriately sized code and broadcast each of the p' packets once. An alternative is to continuously and randomly broadcast packets. For instance, an RSU might continuously broadcast pieces of its certificate revocation list. Another possibility is to divide the fragments between a set of transmitting nodes in the system. Any unit that encounters enough other nodes will be able to reconstruct the message.

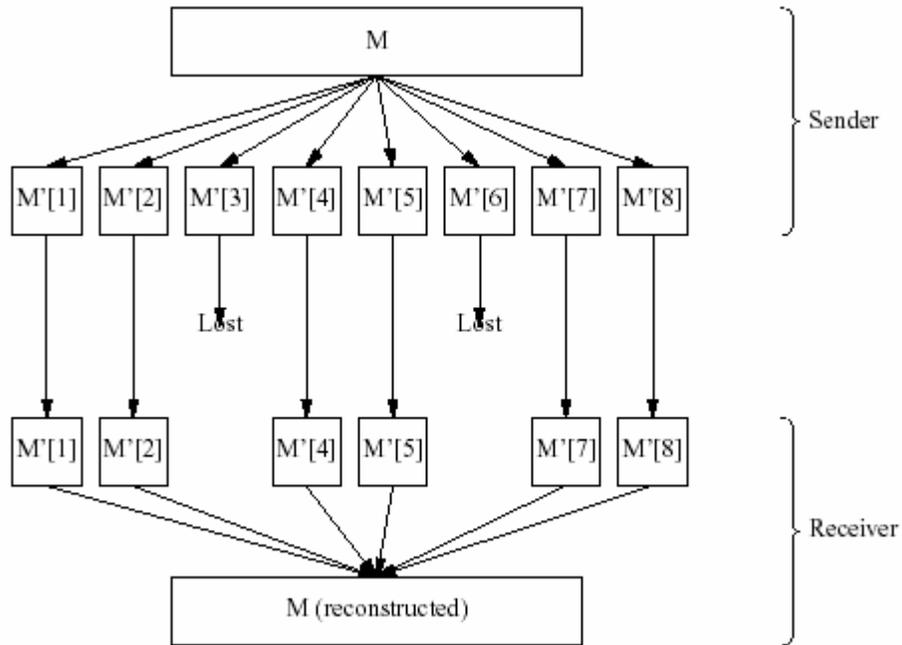


Figure 13: Erasure Code Transmission and Reconstruction

The figure above shows an example. The sender starts out with a message M that is 6 packets long. He then applies the encoding function to break it up into 8 packets. As long as the receiver has any 6 of those packets he can reconstruct the entire message M . In this case, packets 3 and 6 are lost, but the receiver gets the rest and then applies the decoding function to reconstruct M .

The fact that such a function exists is sort of counterintuitive and may be easier to visualize with a simple example. Imagine that we want to transmit a number from 0 to 7. This message can be encoded in three bits.

Value	Code
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

So, imagine that we break it up into three 2-bit messages, as shown in the figure below, so:

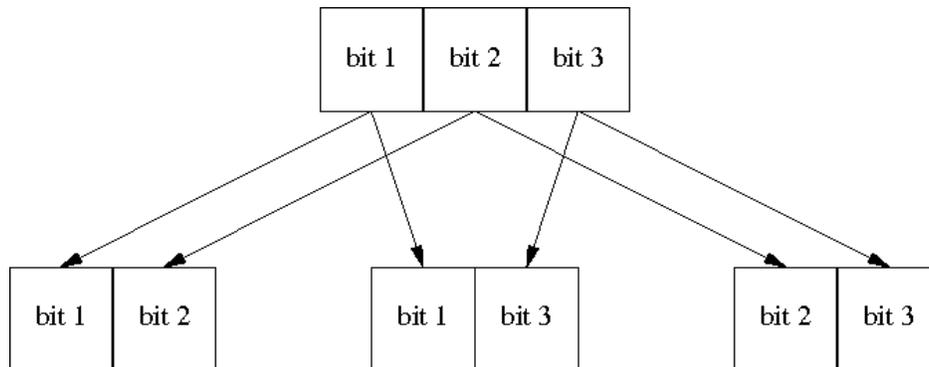


Figure 14: A Trivial Code

Thus, to encode the message 5 (101) we would transmit three packets (10,11,01). A receiver who received any two of these packets (provided he knows which one is which) can reconstruct the original message simply by reversing the encoding process. This example isn't very efficient but should be sufficient to illustrate the principle.

Patent Status

The dominant erasure code choices are subject to a large number of patents; many of which are held by Digital Fountain, Inc. We recommend that any anticipated users perform their own analysis of the patent status of this technology.

Flooding

Another issue for consideration is the distribution of files. Even if files are divided up using an erasure code scheme, the DSRC system is not one big network, but rather a series of small ad hoc loosely connected networks. Data is introduced into the system from a variety of points (typically RSUs controlled by the transmitting agent), and then must propagate to the relevant nodes. The naive solution is to have each node that receives the message rebroadcast it to all local nodes. This is obviously not optimal, and improved algorithms have been extensively studied, for instance in [22, 23]. The optimal flooding protocol for the VSC environment needs to be found/studied.

Once we assume the existence of some flooding protocol, we need to consider that there are two kinds of messages: messages that must be broadcast to all units (revocation lists, for instance) and messages that must be transmitted to only some subset of units (e.g., software updates, which only apply to units from one manufacturer). The first class of messages is *Globally Flooded* using whatever flooding algorithm is finally chosen.

The second class of messages may be transmitted by the *Limited Flooding* algorithm. Each message comes with a label (probably chosen out of a set of 100 or so), and units only listen for messages that fall into their label. For instance, software updates would be labeled by their manufacturer. This creates a set of overlay networks inside the main network each of which gets only the messages directed to it. Note that this leaks some information about unit identity, isolating it to the subset in which it is interested.

This sort of large-scale flooding network is not something that has seen extensive testing in field applications. Therefore, it is possible that there will be unpredicted problems. If such problems occur, it may be necessary to create a network of base stations that distribute data to OBUs. For instance, it would be possible to distribute updates at gas stations, where cars would be able to form a simple reliable network connection with the local RSU and download the update.

4.5.3 Global Symmetric Keys

If we want to preserve privacy, the simple approach is to have every valid OBU use the same key. Since all units have the same key, there are no additional benefits to using asymmetric cryptography. All units can simply use the same symmetric key that is then used by every unit to apply a *Message Integrity Check* (MIC) to each packet. (These are often called *Message Authentication Codes* (MAC) but we will use the term MIC to avoid conflict with the 802.11 MAC layer.) A recipient verifies a packet by recomputing the MIC and then comparing it to the message MIC.

Privacy

A global symmetric key scheme provides excellent privacy. As every message transmitted in the system is MICed with the same key, any given message could have been generated by any OBU in the system, thus providing both anonymity and unlinkability for the authentication portion of the message.

Performance

A global symmetric key scheme would have excellent performance. The only cryptographic operation in the critical path is a symmetric key MIC. MICs are very fast – approximately one million MICs/second were measured on a Pentium II/400. MICs also have extremely low space overhead – a secure MIC can be as small as 80 bits.

Compromise Containment

Compromise containment in such a scheme is handled primarily by changing the global key. Once a unit has been discovered as compromised, a new global key is generated and propagated to all uncompromised units. The key can be distributed using a standard broadcast encryption scheme as described in Section 5.5.2.1. If compromises occur frequently, key changes can be performed periodically rather than upon each compromise. Revocation lists are neither necessary nor useful for a global key scheme.

Because all messages in this system use the same keying material for authentication, it is not possible to revoke a compromised unit based purely on messages that it sends. However, if the unit's private key is published or its identity is known, it can then be revoked. The worst-case compromise scenario for a global key scheme is an attacker who compromises a single unit and then anonymously publishes the global key after every update. Because the global key is the same for everyone, it will be very difficult to track down the attacker or determine which unit was compromised. Such an attacker could potentially compromise the entire system for a long period of time. The VSC team believes that this threat is sufficiently serious that it renders a global key scheme unacceptably subject to catastrophic compromise.

There are variants of this scheme which somewhat limit the scope of compromise. For instance, you could use different keys for different geographic regions. However, because all symmetric schemes require that the sender and the recipient share keying material, they are all vulnerable to catastrophic failure to some extent.

Management Complexity

The management complexity in a global symmetric key system is produced by the need to periodically refresh keys. Any single update has the complexity of a single round of broadcast encryption, as described in Section 5.5.2.1. In the most secure implementation, this complexity would be incurred every time a unit was compromised. However, in practice, key updates would likely occur on the order of every month or so.

4.5.4 Public Key Signature Based Schemes

In order to avoid the catastrophic failure modes described in Section 5.4.3, it is necessary to use public key cryptography of some kind or another. In this section, we describe a number of approaches that use classical public key cryptography. In the next section we describe an approach that uses a newer public cryptographic primitive called "group signatures".

Simple Public Keys

At the opposite end of the spectrum from a single global key, we can issue every OBU its own key and certify it using a PKI. The manufacturer of the OBU would issue these certificates. This scheme would be very similar to that described in Section 5.3 and, as with that scheme, would make use of standard, well-understood cryptographic techniques.

Because the only important authentication in an OBU certificate is the fact that the named public key belongs to a legitimate OBU, OBU certificates can be even simpler than those provided to RSUs or PSOBUs, containing only three data items:

- The public key of the certificate holder.
- The identity of the signer of the certificate.
- A signature over the certificate.

If certificates expire, it may be desirable to have a validity period. However, the difficulties associated with obtaining new certificates suggest that it may be better to control compromise with CRLs only. In practice, however, if such a scheme were adopted it might be desirable to use the same format for both RSUs/PSOBUs and end user OBUs, and simply leave the useless slots empty or filled with some fixed data item. As we shall see, even with this system the privacy guarantees of this scheme are inadequate. We present it as a reference point against which other schemes can be compared.

Privacy

The privacy guarantees of this scheme would be quite limited. Because each message contains the certificate of the signer, it is trivial to link messages sent by the same sender, merely examine the certificate. This property is inherent in the system. One variant that improves the resistance to linkage is to issue multiple certificates to each OBU. If (say) each OBU had 1024 certificates and picked a new random certificate each hour, the ability for observers to link transmissions would be extremely limited. In order to enable compromise containment, however, the CA would need to maintain a list of which certificates have been issued to which OBU – or bind them together cryptographically. This would allow the CA to revoke all the certificates issued to an OBU once one certificate was discovered to have been misused.

This scheme can provide limited anonymity. Because the identity of the OBU is not in the certificate, ordinary attackers cannot directly discover the identity of a vehicle from

its transmissions. Of course, if such an attacker is able to independently determine the identity of the owner of the OBU, they can then link that identity to the OBU's transmissions by observation.

Anonymity against insiders is more limited. Because the manufacturer of the OBU knows which certificates were issued to which units, it is easy to work backward from any given transmission/certificate to determine which unit sent it. In principle, the CA could "forget" which units were issued which certificates, while maintaining information about other certificates were linked to it, however, it is unknown whether consumers will believe such a promise.

Performance

Performance of this system is equivalent to that described in Section 5.3. As with that system, performance is dependent on the underlying signature algorithm. Section 5.3.8 describes the performance characteristics of the candidate algorithms.

Compromise Containment

Containment of compromise in a standard PKI scheme is straightforward. As with the PKI scheme of 4, each bad message contains the certificate of the signer. It is therefore trivial to identify the public key of the responsible unit. As before, once the public key is identified, it can either be placed on a certificate revocation list or the unit can be denied a new certificate once its current certificate expires. Note that both of measures can be taken without the CA knowing the identity of the OBU that has a specific key pair. Similarly, if the private key is compromised and published, it is useless without the certificate. Once the certificate is known, revocation is performed as in the case where a message has been compromised.

Management Complexity

The primary management problem with this scheme is arranging to deliver certificates and CRLs to the OBUs. Unlike the situation in Section 5.3, end-user OBUs do not regularly connect to a single base station and, therefore, cannot get updates from that base station. In order to make this scheme work properly, certificates will need to be distributed by a flooding mechanism, such as that described in Section 5.5.2.3.

The costs for certificates and CRLs are different. Certificates must only be propagated to the appropriate OBU. Therefore, certificates can use a Restricted Flooding system. By contrast, CRLs must be propagated to all units in the system and, therefore, must use a Global Flooding system. The frequency of certificate refresh must therefore be tuned depending on the frequency of certificate revocation.

Systems have been extensively discussed where there is no certificate refresh at all, and all compromise control is done with CRLs. The advantage of such schemes is that all units must receive the same CRL, and so there is no need to partition the certificate updates between relevant units. The disadvantage of this scheme is that if revocation rates are high, CRLs can grow very large. Despite that, it appears that this is a superior approach, especially as it drastically simplifies a number of the other schemes.

Anonymous Certificates

The scheme discussed in Section 4.5.4.1 has two primary problems:

- A high level of linkability for any observer.
- A lack of anonymity against Class 4 attackers.

As discussed previously, the second property is not a necessary security requirement, but merely an artifact of the way that certification works. If the CA could somehow verifiably "forget" the binding between public key and vehicle identity, then it would be possible to construct a system without this privacy problem.

There are three major approaches for allowing certificates to be signed without creating such a binding.

1. Simple administrative policy – the CA promises to forget.
2. Blind signatures – the CA signs a certificate it cannot see.
3. Role separation between the CA and the OEM – the OEM knows the identity but does not communicate it to the CA and is not allowed to see the certificate.

The first option was discussed in the previous section. Although it is possible in principle, there is concern that users will not trust the manufacturer to avoid accidentally creating such a binding. Consequently, the remaining two alternatives are analyzed in greater detail in the sections that follow.

Blind Signatures

Blind signatures [24, 25] allow a signer to perform a digital signature on a message without knowing the message that is being signed. A blind signature on message M works as follows:

1. The client (in this case the OBU) uses a signature algorithm-specific blinding function f and a randomly generated *blinding factor* B to compute $M' = f(B, M)$. He sends M' to the signer.
2. The signer (in this case the CA) signs M' using the ordinary signature algorithm s and his private key K to produce $Sig' = s(K, M')$. He sends Sig' to the client.
3. The client applies the inverse blinding function f^{-1} to compute $Sig = f^{-1}(B, Sig') = s(K, M)$.

This protocol allows the CA to sign a message about which it knows nothing. In the VSC context, the OBU would generate a randomly chosen signature key and have the CA blind sign it. Thus, the OBU would obtain a certificate without having the CA know its identity.

Note that because the signer has no idea what it is signing, blind signature systems are in general vulnerable to attacks in which the client provides a bogus plaintext to the signer for signature. For instance, the OBU might generate an RSU certificate. However, this form of attack is not a concern in this context (though we will need to deal with it in the next scheme) for two reasons. First, OBUs can be initialized at the factory, where the vendor can be certain that they have not been tampered with, and that they will thus provide valid messages for signature. Second, a separate OBU-only signing key can be used for this application. Because the only meaning of such a certificate is that it belongs to an OBU, the scope of attack even for a bad unit would be sharply limited.

Role Separation

An alternative approach would be to separate the authorization to issue a certificate from the certificate issuance proper. Imagine that the CA provides the OBU manufacturer with a "signature box". When that box is physically mated to an uninitialized OBU, it will issue it a certificate. The communication between the OBU and the CA is performed over a secure channel and, therefore, the OBU manufacturer cannot learn the OBU's

certificates. Because the CA does not know the OBU's serial number, it cannot be used to de-anonymize the OBU. This is just another form of administrative control.

Multiple Certificates and Linkage

As discussed in the previous section, in order to prevent linkage of transmissions it might be desirable to issue each OBU a number of certificates. However, because the certificates are blinded, it is no longer straightforward to revoke all of an OBU's certificates when one is compromised. In order to permit this mode of operation, we need to escrow the ability to link certificates. The general idea is that each OBU certificate contains a linkage value W , which can be used to link it to the other certificates issued to that OBU. W is computed as follows:

First, each OBU generates a random symmetric key k . For each certificate i the OBU generates a cryptographically random value $r[i]$ of length w bits.

For each certificate i the OBU computes the following values:

$$W[i] = H(k, i)$$

$$h[i] = H(0 \parallel r[i] \parallel W[i] \parallel \text{LinkageMarkerString})$$

$$e[i] = H(1 \parallel r[i] \parallel W[i])$$

$$B[i] = \text{Encrypt}(e[i], k)$$

Where $H()$ is a cryptographically secure message digest. Upon certificate issuance, the W values are embedded in the certificate, and the $h[i]$, $B[i]$ pairs are provided to the escrow authority.

Thus, when a bad message is identified, the escrow authority can use the certificate's W value to recover the OBU's k , and generate the remaining W values. Note that this operation has been made *intentionally* expensive by the introduction of the $r[i]$ value. Identifying the correct $h[i]$ value requires exhaustively searching all possible values of $r[i]$, which means on average of 2^{w-1} digest operations. This feature increases the cost of linkage, thus increasing privacy. $r[i]$ can be made arbitrarily large, thus making the amount of protection tunable at OBU manufacturing time.

The requirement to solve the $r[i]$ "puzzle" in order to perform this linkage operation is intended to serve as a technical deterrent to casual privacy violations by making linkage expensive.

Note that as long as one of the anonymizing schemes mentioned above is used to hide the identity of any particular OBU certificate, it is not possible to de-anonymize the OBU based upon certificates alone. All that is possible is to identify other certificates belonging to the same OBU.

Privacy

The privacy of this scheme depends on whether escrowed identities are used. If they are not, then transmissions are nearly anonymous, even in the face of Class 4 attackers. However, that means that each OBU can only have one certificate and, therefore, transmissions are highly linkable.

Alternately, if multiple certificates are issued, then transmissions are no longer linkable via certificates, but identity escrowing must be employed to allow revocation of all of an OBU's certificates. The escrow authority can be separate from the OBU manufacturer, but ultimately it is straightforward to determine when two transmissions came from the same OBU (e.g., with a subpoena). However, as long as one of the de-linking schemes

mentioned above is used, it is not possible to de-anonymize the OBU via certificates alone.

Performance

Because this scheme only modifies the certificate issuing process, performance of this scheme would be generally equivalent to that described in Section 5.5.4.1, both in terms of time and space overhead. One difficulty is the choice of signature algorithm. The most common form of blind signatures is for RSA.

Compromise Containment

In general, compromise containment is done in the same fashion as that of Section 5.5.4.1. Once a compromised unit's public key is known it can be revoked.

Because the CA does not see OBU's public keys during the certificate issuance process, OBU certificates cannot be easily renewed once they expire. There are two fixes for this:

- Maintain a global certificate list, and
- Rely solely on CRLs, thereby not employing certificate expirations.

In the first strategy, OBUs would arrange to send copies of their certificates anonymously to the CA. The CA can verify that the certificates are validly signed and, if the public key has not been revoked, issue a new certificate, which is then distributed via the flooding mechanism of Section 5.5.2.3.

Without certificate expiration, CRLs may quickly grow quite large, however at the end of the day it may still be simpler to choose this approach.

Management Complexity

This scheme may introduce new management problem of maintaining the global certificate list. However, because each OBU only needs to deliver its certificate to the CA once, the problem is not particularly severe. In general, the easiest way for this to work is to have a modest number of RSUs that capture certificates and forward them onto the CA, which maintains the list. Since every message is sent with a certificate, no special cooperation is required from the OBUs.

If no global certificate list is maintained, this scheme has the same management complexity as ordinary certificates.

Anonymous Self-Enforcing Certificates

The previous two schemes allowed for linkage by the identity escrow agent. Thus, there is a risk that the identity escrow agent can be used to link any set of transmissions. Worse yet, if the identity escrow agent keeps records, then it can be used to de-anonymize transactions. We describe a complicated scheme for preventing such attacks here.

In this system, each OBU U_i has a long-term symmetric key K_i . The CA maintains a global mapping of OBU identity to K_i . In order to get a certificate, an OBU must engage in an interactive protocol with the CA. Logically, this protocol has four stages:

1. U_i demonstrates possession of K_i to the CA, perhaps by signing an initial challenge.
2. U_i generates k key pairs I_1, I_2, \dots, I_k .
3. The CA blind-signs certificates for all k keys. Each key has a one-day validity window.

4. U_i un-blinds the certificates.

After this exchange, U_i has k days worth of certificates. It uses one certificate per day until all the certificates have expired, at which point it must re-run the protocol to get a new batch of certificates.

Because the certificates are blind-signed and no longer contain U_i 's long-term identity, it is no longer possible to revoke them based on a single bad message. In order to allow for compromise containment, we construct the key pairs so that they are self-enforcing. In DSA, ECDSA, and BLS, the private key X is 160 bits long and randomly chosen and the public key Y is computed from X . Instead of choosing X randomly, we choose it so that the top 40 bits contain the identity i of the OBU. Thus, if a private key is compromised, the identity can be extracted and no future certificates issued to U_i .

In order to ensure compliance by the OBUs (who might not put their identities in the certificates) the protocol must be more complicated. The cut-and-choose [34] technique is used to enforce OBU behavior. Thus, we replace steps 2 and 3 above with:

1. U_i generates $2k$ key pairs.
2. U_i generates $2k$ unsigned certificates, 2 for each of the k periods, using the $2k$ keys generated in step 1. Call them $C_1^1, C_1^2, C_2^1, \dots, C_k^1, C_k^2$.
3. U_i blinds each candidate certificate to get C'_1^1, C'_1^2, \dots and sends the blinded certificates to the CA.
4. The CA randomly chooses one of each pair (C'_j^1, C'_j^2) and asks U_i to reveal that blinded certificate.
5. U_i sends the CA the original C_j^l 's values, their blinding factors, and the corresponding private keys.
6. The CA verifies that the unblinded certificates match their blinded counterparts and that the private keys correctly contain i .
7. If all these things are true, the CA signs the other certificate in each pair and returns it to U_i . Otherwise, it adds U_i to the compromised unit list.
8. U_i unblinds the signed certificates.

This procedure is designed to minimize the probability that an OBU can get a certificate without its identity embedded. Consider what happens if the OBU omits its identity. Without loss of generality, assume that C_1^1 is improperly formatted. There is a 50% chance that the CA will ask O_i to disclose C_1^1 , in which case O_i will be caught. The other half the time, the CA will choose C_1^2 in which case the attacker will have a single key, which he can use for one day without revealing his identity.

If the attacker wants to obtain more certificates with fake identities, the chance of success drops dramatically, halving with each additional false certificate requested. Thus, the probability that an attacker can obtain any significant number of untraceable keys is very low. This attack is dominated by simply compromising a unit that has just completed the key exchange protocol and, therefore, has k valid (though traceable) keys.

Privacy

As with the previous scheme, this scheme provides full anonymity for the certificates. Although the identity of the unit is known to the CA at the time of certificate issuance, there is no way to map this information to the certificate itself, as the certificate was

signed with a blind signature. Messages can still be linked within the lifetime of a single key but, as that lifetime is short, the severity of linkage is severely reduced.

Performance

The performance of messaging using this scheme is the same as that of the previous two PKI-based schemes. The management performance is worse, however, as discussed below.

Compromise Containment

Compromise containment in this scheme is worse than in the previous two schemes. First, the system cannot revoke certificates. Because the certificates are blindly signed, there is no way to determine which certificates are to be revoked. The only recourse is to not issue new certificates to the compromised unit in the future. It is not clear how severe this limitation is. If investigations take weeks to months, then the time from identification of the bad unit to key expiry might be small compared to the investigative time period.

Second, the system can only contain compromises in the second two of the three cases described in 4.5.1.3. If the private key is published, the system can use the self-enforcement property to determine the identity of the offender and not issue him future certificates. Similarly, if the system knows the identity i of the unit, it can simply refuse to issue it certificates in the future.

However, if the only information that is available is some set of messages generated by the compromised unit, the scheme as described provides no way to contain compromise.

Management Complexity

This scheme is much more complicated to manage than the previous two schemes. In particular, it requires a periodic interactive exchange between OBUs and the CA. This exchange consists of two round trips with sizes shown in Table 4 below.

Sender	Message	Size
OBU	C 's	$2kS_{sig}$
CA	request for C values	k bits
OBU	revealed C	$k(S_{pub} + values S_{sig})$
CA	signatures	kS_{sig}

Table 4: Message Sizes for Blind Signatures

In the best-case scenario of a signature and key size of 20 bytes and $k = 30$, the OBU must send 2400 bytes to the CA and the CA must respond with 600 bytes. Moreover, this is a point-to-point (not broadcast) communication, so it would be necessary to design a mechanism for OBUs to communicate reliably with CAs, which the previous schemes did not require. Because the OBUs will generally not be within direct radio contact with the CA, the DSRC network will need to provide a method for routing traffic through to the CA, probably by routing to a connection to the Internet. This requires Internet gateways in a large number of locations.

Static Combinatoric Schemes

Another approach is to use an adaptation of the cover-free broadcast encryption scheme described by Garay et al.[35]. This modification was first proposed by Jiang [36]. We have a set \mathbf{R} , consisting of R key pairs and N vehicles. All public keys are known to all vehicles, and each key is assigned an index l . Each OBU U_i is assigned some random subset S_i of \mathbf{R} containing n keys. We assume that vendors know which keys were assigned to each OBU, which seems like the likely case.⁷

When a unit U_i wishes to send a message M it picks a key j (for the moment, assume that the transmitting key is chosen randomly and held constant until revoked) It then signs M with R_j and then transmits the signed message $M, \{M\} R_j, j$. To verify, the receiving unit looks up key j in its local key table and verifies the signature.

Compromise is handled by *compromised key lists* (CKLs), which are globally broadcast. Whenever a bad message is detected, the key with which it was signed is added to the CKL. Units must verify that a given key is not on the CKL before accepting a message. Note that because any given unit has n keys, at minimum n bad transmissions from that unit must be observed before it can be revoked. If the attacker uses each key for the duration of the investigation, then he can use each compromised key for n investigatory periods.

The properties of this scheme depends on the chosen parameters R and n as well as the size of the system N . N is an exogenous parameter set by the number of vehicles in the system. For our purposes, we use N of 10^9 (4 times the number of vehicles in the US). Given these parameters, we can easily determine:

- The probability that any given vehicle will have a key R_l is n/R . The average number of vehicles with a given key is Nn/R .
- The probability that any given vehicle is currently using R_l is $1/R$. Therefore, the average number of vehicles using any given key R_l is N/R . Call this value C .
- Once r keys have been revoked, the probability that a given innocent unit has been covered completely (revoked) is approximately $(r/R)^n$. Thus, on average, in a system of size N , $N(r/R)^n$ units will be completely revoked.
- The number of possible key subsets is approximately R^n if $n \ll R$.

In order for this system to work correctly, at minimum it would be preferable that no two units, U_x and U_y , have the exact same key subset. Otherwise a revocation of U_x will automatically revoke U_y and this is very undesirable. The probability of a collision is about 50% if the number of subsets is N^2 . For safety, say that the number of subsets should be $10 \times N^2$, which results in $R^n = 10^{19}$ for $N = 10^9$.

Privacy

The privacy behavior of this scheme depends on the settings of n and R , which so far have only been partly constrained. In order to minimize linkability, it must be reasonably likely that two observed messages signed with the same key were not generated by the same unit. The definition of "reasonably likely" is fuzzy; semi-arbitrarily, the team deems a 50% probability that the message was generated by the same unit as "reasonably likely".

⁷ It is not clear how to design a system that hides this information cryptographically, as opposed to with administrative controls.

Say that the system has observed a message M with key R_i , and U_i is currently known to be using key R_i (e.g. via a previously observed message signed with R_i), then the probability that U_i generated M has as its upper bound $1/C$. In practice, the probability is far higher than this since vehicles are not homogeneously distributed and tend to stay in given areas. In a closed area of size A , the probability becomes $N/AC = R/A$. In order to satisfy our linkability requirement, then, we require that $R/A < 0.5$. If we assume that the smallest area we are concerned with is 10^5 people, then $R \leq 2 \times 10^5$. Since $R^n = 10^9$, the minimum value for n is 4.

One alternative is to have units choose random keys out of their available key set rather than use the same key for a long period of time. In this case, given that a unit U_i has known key R_i and the system then observes a unit using that key, then the probability that that unit is U_i is R/Nn . Again, partitioning the world into geographic areas results in the probability of that increases to AR/Nn . If $n = 4$, then R is potentially as high as 4×10^5 . Note, however, that if a fresh key is chosen for each transmission, then a very small number of transmissions are enough to uniquely identify the unit, since it will quickly cycle through all its keys. Thus, each key must be retained for some moderate period of time, during which linkage is easy.

Performance

Both time and space performance of this scheme is excellent. The CPU cost is the usual cost of a digital signature. The space cost is the cost of the signature itself. Because keys are identified by indices, there is no need for certificates, thus the space overhead is roughly half that of the PKI schemes. Verification overhead is also improved because there is no need to verify a certificate, even on the first interaction.

Compromise Containment

The compromise containment scheme described here has two problems:

1. It requires that a large number of bad messages (one for each key in the unit) be observed before any given OBU is completely revoked. For instance, if $n=5$, a Class 3 attacker who had compromised the unit could mount 5 major attacks before his unit is revoked.
2. It is brittle when the number of compromised units is not very small.

The first problem is obvious, but the second requires further discussion. Recall that once r keys have been revoked, the number of revoked innocent units is approximately $N(r/R)^n$. Thus, after u units have been revoked, the number of revoked innocent units is approximately $N(un/R)^n$. The figure below shows the number of revoked innocent units for $R = 100,000$, some plausible values of n , with $N = 2.5 \times 10^8$, which is approximately the number of active vehicles in the United States.

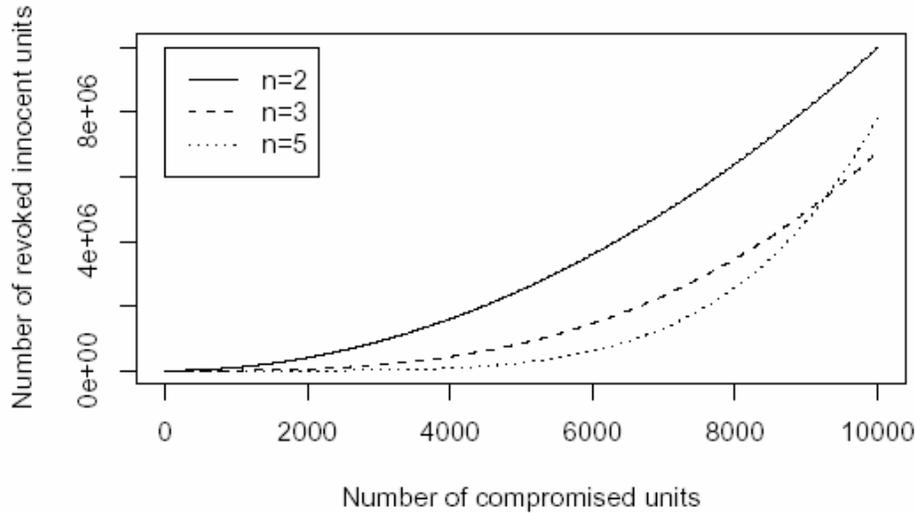


Figure 15: Innocent Revocation Rates

As is apparent above, when even small fractions of the total number of units are revoked (10,000 is only 0.004% of vehicles), the number of innocently revoked vehicles becomes completely unmanageable. Thus, a mechanism for re-keying is required. This result is not unexpected and parallels that seen by Garay et al. [35]

Management Complexity

If re-keying is not used, then management of the system is very straightforward. The only management required is distribution of CKLs, and these can be distributed by the flooding mechanism of Section 4.5.2.3. However, the management complexity of re-keying is potentially highly problematic. This topic is discussed in the next section, entitled, dynamic combinatoric schemes, which requires frequent re-keying.

Dynamic Combinatoric Schemes

Given that a re-keying mechanism is necessary for any re-keying scheme, an alternative approach is to use a dynamic combinatoric scheme with frequent re-keys, as follows. To issue keys for K time periods, generate K universes of key pairs, $\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_k$, with S keys in each set \mathbf{R}_k . Thus, within each universe \mathbf{R}_k are keys $R_k^1, R_k^2, \dots, R_k^S$. For each time period k , the system generates S disjoint subsets of N, N_1, N_2, \dots, N_s such that \cup from i $N_i = N$. A unit's membership could be represented as a list of set indices. For instance, $(1,5,7)$ means that the unit was in R_1 in time period 1, R_5 in time period 2, and R_7 in time period 3. In each time period k , all members of N_i are assigned key R_k^i . Thus, the example unit above would be given keys (R_1^1, R_2^5, R_3^7) . As with the previous scheme, each unit gets all the public keys.

As with the static combinatoric scheme of Section 4.5.4.4, authentication is by simple digital signature. Each unit has exactly one key for any time period k and it uses that key for signature during that entire time period. If the original message is M and the unit has key R_k^i then the authenticated message is $M, i, \{M\}_{R_k^i}$.

Upon receipt, the recipient looks up the appropriate public key and verifies the digital signature.

Because each key is only used for a short period of time, this scheme has no support for certificate revocation. Instead, traitor tracing does revocation. Once a series of bad messages is observed, the issuing authority can figure out which unit was assigned those keys, and then refuse to issue future keys to those units. It thus takes at most k time periods from the discovery of compromise to cutting the compromised unit out of the system.

Privacy

As with the static combinatoric schemes, the privacy guarantees for this scheme depend on the system parameters, in this case S . The number of units who are using key i at any given time is N/S . Recapitulating the linkage computation from Section 5.5.4.4, the requirement becomes that $A/S > 2$ – the total number of key subsets must be smaller than the total number of vehicles in an area. For areas of $100,000$, this results in $S \leq 50,000$.

Note that observation of transmissions from the same unit on multiple days quickly allows determination of the unit's true identity. In fact, this technique would likely be used to trace compromised units. However, this is of no help in linking it to future transmissions since it (almost) always shares keys with other units in the same area.

Performance

Both time and space performance of this scheme are excellent. The CPU cost is the usual cost of a digital signature. The space cost is the cost of the signature itself. Because keys are identified by indices, there is no need for certificates, thus the space overhead is roughly half that of the PKI schemes. Verification overhead is also improved because there is no need to verify a certificate, even on the first interaction. Alternately, the issuing authority can provide group certificates to key holders, thus making the message size the same as with a PKI system.

Compromise Containment

Purely refusing to issue future keys to compromised units achieves compromise containment in this scheme. Once a critical number of messages from a compromised unit have been observed (on multiple days), the identity of the unit can be determined and no future keys issued to that unit. Note that because each key is only valid for a very short time window, the attacker is forced to use multiple keys if he wants to mount a large number of attacks, thus leaking large amounts of information and enabling investigation and compromise containment.

Note that in the worst-case scenario, compromised key lists can be used to provide fast update once an offender is identified. However, given that in general the window of vulnerability is a month or so, it is doubtful that this is worth doing.

Management Complexity

The management complexity of this scheme is primarily in the need to broadcast the periodic key updates. Consider the size of an update for a single time period. This update has two parts:

1. S public keys.
2. S keys encrypted for N recipients (N/S recipients per key).

The first part of the update is of size approximately $S_{pub} S$ (1 megabyte for 20 byte public keys) and can be distributed by simple Global Flooding. Alternately, group certificates can be provided to OBUs along with their private keys.

The private key update is more difficult. Naively, it requires N separate encryptions, one for each OBU. However, this creates an unworkably large number of messages that must be transmitted. If the ciphertext size is the same as the key size S_{pub} , then the total ciphertext size is $S_{pub} N$ (approximately 2×10^{10} bytes for $N = 10^9$ and 20 byte keys. This would need to be distributed by Limited Flooding, but is still quite expensive (about 10^{11} bytes per month total).

A superior approach is to use a broadcast encryption scheme, such as that described in Section 4.5.2.1. Assuming a very high revocation rate of 1%, then the ciphertext size for each key is $0.02 \text{ times } N/S$ (8000 bytes for 20 byte private keys), with a total size of $4 \text{ times } 10^8$ for each time period (10^{10} bytes for a month). Ciphertext size scales linearly with the number of revoked keys, so for a more reasonable revocation rate of 0.01%, the total size would be a mere 10^8 bytes per month.

In order to enable this scheme, each unit would be constructed with its share of the broadcast encryption trees for its entire lifetime built into the unit. Each broadcast encryption tree share is of size $1/2 \log^2 N$ keys. For ordinary 128-bit symmetric keys, even if we change keys every day, 20 years worth of keying material can be stored in 25 megabytes of permanent (read-only) storage. In practice, it is unlikely that it is necessary to have either this many separate groups or change keys this frequently. In practice, a few hundred different permutations chosen at random is probably sufficient.

4.5.5 Group Signatures

The final approach to consider is *group signatures*. Group signatures are a relatively new cryptographic primitive originally proposed by David Chaum [37]. A group signature scheme has a single public key P and a large number of private keys K_i . Each member of the group is issued a single private key, which can then be used to generate signatures that verify with P . Outsiders can only verify that a signature was generated by *some* member of the group but cannot tell which member.

In general, group signature keys are distributed by a central authority. That central authority can determine which key pair generated any given signature. Although it is possible to hide the identity of any given key holder from the central authority, this capability allows the central authority to link multiple signatures by the same signer. When a key is determined to have been compromised, the central authority propagates an update to each verifier, which allows them to exclude signatures by that key.

The best published scheme is by Ateniese et al [38], and is generally known as ACJT. Unfortunately ACJT signatures are approximately seven times the size of comparable strength RSA signatures. This makes ACJT unsuitable for this purpose. Boneh and Shacham are currently devising a superior group signature scheme, that may have comparable signature size to RSA, however, it has not yet been published or widely reviewed. General security practice very strongly discourages the use of new algorithms and, therefore, the VSC team cannot recommend this algorithm for deployment in any system until it has had a year or so of public analysis. Depending on the VSC timeline, this may or may not be an attractive option.

4.5.6 Comparison of Schemes

Seven schemes have been heretofore described, however, Global Symmetric Keys and Group Signatures can be excluded immediately. Global Symmetric Keys provide unacceptably poor containment of compromise and Group Signatures provide unacceptable performance, primarily in terms of message size. Similarly, Static Combinatorics can be excluded because Dynamic Combinatorics dominates it.

The remaining schemes are:

- Simple Public Keys
- Anonymous Certificates
- Anonymous Self-Enforcing Certificates
- Dynamic Combinatorics

The choice between these four comes down to a tradeoff between privacy, management complexity, and containment of compromise.

Simple Public Keys offers the lowest management complexity, however it provides only very weak privacy protection. In the best-case scenario, where OEMs do not keep track of the binding between OBU certificates and vehicle identity, anyone who obtains access to the CA linkage database (however maintained) can use it to construct a vehicle tracking system. Once a vehicle is observed once, all of its certificates could be identified and, thus, tracking is easy. In the worst-case scenario, if the OEM maintains a vehicle/certificate binding, a vehicle can be identified from a single transmission.

Anonymous Certificates offer a slightly larger management complexity, but if certificates are never reissued (only CRLs are used) this complexity is confined to the certificate issuance process. They have similar linkage properties to Simple Public Keys but offer superior protection against the identification of vehicles -- as opposed to certificate revocation -- from a transmission. As the no-re-issuance certificate model seems likely, the VSC team believes that the increased public confidence and auditability of these schemes justifies the additional management complexity at certificate issuance time. Role separation appears to be a superior option for implementing Anonymous Certificates. The complexity of certificate re-issuance strongly suggests that having certificates not expire at all and handling compromise purely through CRLs is the best approach, although some additional work needs to be performed to identify schemes to bound the size of the CRL as the number of OBUs grows very large.

Anonymous Self-Enforcing Certificates provide superior privacy; they are not susceptible to the de-anonymizing attack described above. However, they provide inadequate compromise containment because they cannot be revoked based on transmissions but only on private key leakage. Because of these considerations, the VSC team considers Anonymous Certificates to be a superior alternative.

The only plausible alternative to Anonymous Certificates is Dynamic Combinatorics. Unlike Anonymous Certificates, it cannot be turned into a tracking system because multiple OBUs always have the same public keys. However, Dynamic Combinatorics have a much higher management load than Anonymous Certificates in that they require regular updates of keying material that must be delivered precisely to specific OBUs (rather than globally broadcast). Moreover, the updates are much larger than CRLs are likely to be. There are also concerns that this scheme would be unworkable in practice. Accordingly, the design team recommends the Anonymous Certificates approach as offering the best combination of security, privacy, and reliability. However, given the benefits and liabilities of both of these schemes, and the potential privacy risks associated with certificate linkage, we feel it would be wise to subject these schemes to a detailed assessment in the context of the large vehicle population and the well understood vehicle life cycle, and retirement process. In addition it may be advantageous to explore some combination of these schemes to manage the size and growth of the CRL, while addressing the compromise of privacy and the management of the large number of retired OBUs.

5 Protocol

5.1 Introduction

The previous section described potential architectures and provided a recommendation from the team. This chapter provides a complete description of VSC Security Protocol (VSP), the enrollment procedures, suggested policy requirements, OBU privacy parameters, opportunities for optimization, and a summary of the syntax.

5.2 Protocol Summary

With the identification and description of a security architecture that appeared to meet the needs of vehicle safety applications (see Chapter 4), a security protocol was able to be created, based upon this identified architecture. Whereas the security architecture described the cryptographic skeleton, the protocol specification describes the contents and formatting of each protocol message, as well as the behavior of each communicating endpoint. The protocol specification could therefore serve as a basis for the eventual implementation of the system. The architecture and protocol will be used to validate the requirements detailed earlier in this report by providing the DSRC standards writing group with a feasible solution. This chapter describes the security protocol that was developed in detailed protocol definition language, and ends in Section 5.8 with a complete summary of the syntax described throughout the chapter.

5.3 Protocol Definition

Security protocol is a structured approach toward providing descriptions of technical details. For clarity, the protocol is described using the protocol definition language used by TLS [43]. This section begins by describing the certificate and CRL format in the standard protocol definition language, followed by structured descriptions of the message format and a mechanism for propagation of system updates. Finally, it describes the procedures required for enrollment of new units in the system.

5.3.1 Certificate Format

Before units can authenticate to each other, they must first be issued credentials in the form of certificates. This section describes the VSP certificate format. In general, the team preferred not to invent a new certificate format, but the requirement to keep message sizes small ruled out the use of conventional certificate formats such as X.509. All VSP certificates use the same basic structure, as shown below:

```

struct {
    uint8      certificate_version;
    SubjectType  subject_type;
    CertSpecificData  type_specific_data;
    Date        expiration;
    CRLSeries   crl_series;
    SignerID    signer_id;
    PublicKey   public_key;
} UnsignedCertificate.

opaque[8]    SignerID;
uint16      CRLSeries;

struct {
    select(subject_type){
        case ca:
            CAScope  scope;
        case rsu:
            RSUScope  scope;
        case psobu:
            PSOBUScope  scope;
        case obu:
            LinkageData linkage;
        case crlsigner:
            CRLSeries  responsible_series<2^16-1>;
    }
} CertSpecificData;

enum {ca(0), rsu(1), psobu(2), obu(3), crlsigner(4), (255)} SubjectType;

uint16 Date;

struct {
    SignatureAlgorithm algorithm;

    select(algorithm){
        case ecdsa:
            ECDSAKey key;
    }
} PublicKey;

```

```

struct {
    opaque point<1..2^8-1>;
} ECPPoint;

enum { ecdsa(0), (255)} SignatureAlgorithm;

struct {
    uint16      length;
    UnsignedCertificate unsigned_certificate;
    Signature    signature;
} VSPCertificate

opaque Signature<1 .. 2^16-1>;

```

The `certificate_version` field contains the version of the certificate format. In this version of the protocol it is 1. Note that this applies to both `UnsignedCertificate` and the `VSPCertificate` structure. Accordingly, when parsing a certificate the recipient must first read the version number in order to know how to parse the remainder of the certificate.

The `subject_type` field describes for what kind of entity the certificate is. Permissible types are `ca(0)`, indicating a CA, `rsu(1)`, indicating a RSU, `psobu(2)`, indicating a PSOBU, `obu(3)`, indicating an OBU, and `crlsigner` indicating that the holder of the certificate is authorized to sign CRLs.

The `expiration_date` field contains the last date on which the certificate is valid. This is represented in days since the UNIX epoch. Expiration occurs at midnight GMT.

The `type_specific_data` field contains information that is unique to this certificate type.

The `crl_series` field contains an integer that represents which of multiple CRLs maintained by the CA this certificate will appear on if it is ever revoked. `crl_series` only have meaning within the context of a given CA.

The `signer_id` field contains the identity of the certificate that the CA is using to vouch for this certificate. For certificates that are not trust anchors, the `signer_id` field must contain the high order 8 bytes of the SHA-1 hash of the CA's certificate. This allows for order 2^{38} CAs to exist before there is an appreciable probability of hash collision. See Section 4.4.2 for information on trust anchor certificates.

The `public_key` field contains the public key of the subject of the certificate. The `PublicKey.algorithm` field contains the algorithm for which the public key is to be used. The only currently defined algorithm is `ecdsa(0)`, corresponding to the Elliptic Curve DSA algorithm specified in X9.31. Section 5.3.1.1 describes the format for elliptic curve public keys.

The `length` field of the Certificate structure contains the length of the remaining structure (i.e., the combined length of the `unsigned_certificate` and `signature`.)

The `signature` field contains a digital signature over the encoded `unsigned_certificate` value.

ECDSA Public Keys

In order to specify an ECDSA public key, the protocol needs to first specify a curve and a point. This is done in the following structure:

```
struct {
    NamedCurve    curve;
    ECPoint       point;
} ECDSAKey;
```

The “Curves” of the Key

Although in principle each key can have its own curve, in practice it is more efficient to select a small number of curves. This approach also has the benefit that certificates are smaller, since they do not need to contain the curve description, but only a curve identifier of type `NamedCurve`. NIST has chosen some curves that are suitable for use in Federal applications, and a subset of those curves is supported here. Additional curves can be registered in the future if needed.

```
enum {
    K-163(1),
    B-163(2),
    K-233(3),
    B-233(4),
    reserved (5..239),
    private (240..255)
} NamedCurve;
```

The `enum` names indicate the corresponding curve in the Recommended Elliptic Curves for Federal Government Use, dated July 1999. Values 240 through 255 are reserved for private use and will not be registered.

The “Point” Of The Key

The other part of an ECDSA key is the point (the public key itself). ECDSA points are expressed in the compressed format of ANSI X9.63 [50].

```
struct {
    opaque point <1..2^8-1>;
} ECPoint;
```

The ECPoint contains the byte-string representation of the point using the conversion routine of Section 4.3.6 of ANSI X9.62 [46]. All points are in a compressed representation.

CA Certificates

In addition to the standard certificate data, a CA certificate contains a scope field of type CAscope:

```
struct {
    ApplicationID    applications<0 .. 2^16-1>;
    SubjectType     unit_types<1 .. 2^8-1>;
    GeographicRegion    region;
} CAscope.

struct {
    ApplicationType    type;
    ApplicationSubtype    subtype;
} ApplicationID;

enum {cert_transmit(0), crl_transmit(1), (2^16-1)} ApplicationType;
opaque ApplicationSubtype<0 .. 255>;
enum {fragment(0), (2048)} MessageFlags;
```

The applications type indicates the application types for which this CA can issue certificates. If this field is empty, it indicates that the CA can issue certificates for any application. This type is structured.

The type field contains the major application type. This is a 16-bit field, thus allowing 65535 possible applications types. The VSCC will maintain a registry of different application types. Application types from 0xf000-0xff00 are reserved for private use but may be transmitted by production units. Types beginning with 0xff are reserved for private test networks and must not be sent by production units. Implementations must ignore message types they do not recognize.

The subtype field provides application-specific information about the message type. This data is not parsed by the VSP protocol but is assumed to be passed up to the application program. The intent is to allow certificates and messages to contain finer-grained control information than is appropriate here. For instance, application X might allow multiple message types but some units might only be able to send some of those types. All of the application types defined in this document have zero-length subtype fields.

The unit_types field lists the unit types (CA, RSU, PSOBU, etc.) for which this CA can issue certificates. If the unit_types field is empty, then it indicates that this CA can issue certificates for any unit type. This usage is not recommended.

The section “region” indicates the area in which the CA is allowed to issue certificates. It is encoded using the `GeographicRegion` type.

Encoding a Geographic Region

In VSP geographic regions are encoded using a `GeographicRegion` structure:

```
struct {
    RegionType region_type;

    select(region_type){
        case polygon:
            PolygonalRegion polygonal_region;
        case circle:
            CircularRegion circular_region;
        case rectangle:
            RectangularRegion rectangular_region<2^16-1>;
    }
}

enum {polygon(0), circle(1), rectangle(2), (255)} RegionType;
```

VSP supports a number of different region types. Currently, three are defined: polygon (0), circle (1), and rectangle (2). All region types with values < 240 must be assigned before usage. Values 240-255 will not be assigned and are reserved for private usage in test environments. Production units must not generate messages with these types:

```
PolygonalRegion    2DLocation<2^16-1>;
```

The `PolygonalRegion` type defines a region as a series of geographic points. The region is specified by connecting the points in a connect-the-dots arrangement in the order they appear. The final point is connected to the first point. The implied lines that make up the sides of the polygon must not intersect. Receiving units should verify that lines do not intersect and reject any region that does. The allowed region is the interior of the polygon.

```
struct {
    2DLocation center;
    uint16 radius;
```

```
} CircularRegion;
```

A `CircularRegion` is simply a circle with “center” at its center and a radius of “radius” meters. The allowed region is the interior of the circle.

```
struct {  
    2DLocation    upper_left;  
    2DLocation    lower_right;  
} RectangularRegion;
```

A `RectangularRegion` is a rectangle formed by connecting in sequence:

1. (upper_left.latitude, upper_left.longitude)
2. (lower_right.latitude, upper_left.longitude)
3. (lower_right.latitude, lower_right.longitude), and
4. (upper_left.latitude, lower_right.longitude).

Note that the `rectangular_region` value is an array of `RectangularRegion` structures. This is interpreted as a series of rectangles. The permitted region is any point within any of the rectangles.

RSU Certificates

RSU certificates contain two scope indicators. First, as with CA certificates, they contain the geographic region in which the RSU is permitted to operate. Second, they contain an indication of which types of application the RSU is permitted to send messages for:

```
struct {  
    GeographicRegion    region;  
    ApplicationID        applications<0 .. 2^16-1>;  
} RSUScope;
```

The scope parameter is interpreted the same was as for CAs. The applications parameter contains a list of the `ApplicationIDs` that the RSU may generate. The RSU must not use this certificate to sign messages application IDs not on this list. If a recipient detects such message, it must reject it. Note that a given RSU may have multiple certificates with different application or geographic scopes. For instance, a traffic signal RSU might be enhanced to provide a maximum speed warning. It may be simpler to issue a second certificate rather than reissue the initial one. Whether to issue multiple certificates or a single certificate with multiple application IDs is a local policy issue.

PSOBU Certificates

The scope field of PSOBU certificates is effectively the same as the scope field of RSU certificates:

```

struct {
    GeographicRegion    region;
    ApplicationID       applications<0 .. 2^16-1>;
} RSUScope;

struct {
    GeographicRegion    region;
    ApplicationID       applications<0 .. 2^16-1>;
} PSOBUScope;

```

Note that it is quite likely that a PSOBU will have multiple certificates corresponding to different geographic regions. This is a straightforward way to provide permissions for a number of disjointed regions or to tile regions with a complicated boundary. The size and placement of certificate scopes is a local policy matter for the CA.

OBU Certificates

OBU certificates have some significant differences from other types of certificates. First, unlike CA, RSU, or PSOBU certificates, OBU certificates are all alike. The only meaning of an OBU certificate is "this public key belongs to an OBU". Thus, OBUs do not have any scope value, as the only important authorization is communicated by the fact that it is a certificate of type obu.

OBU Certificate Linkage

The anonymity of certificates presents a problem for revocation. Once a given certificate is implicated in a bad act, revoking it is straightforward. However, because each OBU has a large number of certificates, this does not substantially contain the compromise of a unit, as the attacker can simply use another one of the unit's certificates. Thus, it needs to be possible to revoke all of a unit's certificates when one is observed. The linkage field allows this mapping to be performed.

Because the ability to revoke all certificates issued together is equivalent to the ability to link transmissions from a single unit, it must be impossible for generic attackers to perform this operation--the ability to link certificates is confined to a trusted identity escrow agency. This section describes a linkage token with the appropriate properties.

We assume that the escrow authority has an ECDH key pair where the public key Y and group \mathcal{G}, \mathcal{P} is known to the OBU at certificate issuance time. We also assume an "anonymity work factor" w and that each OBU is issued n keys. Those keys are numbered from 1 to n .

1. The OBU generates a cryptographically random key k of length 256 bits.
2. For each certificate i (where i is the key number) the OBU generates a cryptographically random value $r[i]$ of length w bits.
3. For each certificate i the OBU computes the following values:

```

W[i]=AES-ECB(k,i)
h[i]=SHA1(0||r[i]||W[i]||LinkageMarkerString)
e[i]=SHA1(1||r[i]||W[i])
B[i]=AES(e[i],k)

```

where `LinkageMarkerString` is the ASCII representation of "VSP 1.0 OBU Linkage Computation".

The encryption of `B[i]` is performed using the AES key wrap defined in RFC 3565[44] 128-bit AES. The first 16 bytes of `e[i]` are used as the AES key.

The `h[i], B[i]` pairs are encrypted under the escrow authority's public key. The exact mechanism is a local matter, however we recommend S/MIME.

The linkage value for certificate `i` is given as:

```

struct {
    opaque    enc_w<2^8-1>;
    uint16    i_value;
} LinkageData;

```

`enc_w` is simply the high order bytes of `W[i]`. Currently, implementations should use 10 byte linkage values.

`i_value` contains the `i` value for this specific certificate.

When an OBU certificate is implicated in bad actions, we wish to be able to revoke all the certificates issued to that OBU. In order to do that, we need `k`. Given a `LinkageData` value, the escrow authority can determine `k` using the following algorithm.

1. Decode the structure to recover `W[i]`.
2. Exhaustively search candidate `r[i]` values until one is found that produces a known `h[i]` value. Note the use of the `p` in the `h[i]` computation. This is intended to assist in the detection of attempts to use distributed computation networks to perform this operation.
3. Form `e[i]` and decrypt `k`.

In order to revoke a unit the CA simply publishes its `k` value on the CRL. Any receiver can easily compute the relevant `w[i]` values.

The choice of `w` is a tradeoff between work to recover from compromise and privacy protection.

CRL Signer Certificates

A certificate of type `crl_signer` indicates that the holder of this certificate is entitled to sign CRLs for this certificate authority. The `responsible_series` array contains all of the CRL series for which this CRL can sign.

5.3.2 Certificate Revocation List Format

VSP uses two kinds of CRLs, one for non-anonymous units such as CAs, RSUs, and PSOBUs and one for anonymous units such as OBUs. However, both CRLs have the same basic structure, which combines both regular CRLs and delta CRLs.

```
struct {
    uint32      length;
    UnsignedCRL  unsignedCRL;
    Signature    signature;
} OrdinaryCRL;
```

```
struct {
    CRLSeries    crl_series;
    SignerID     certificate_type;
    SubjectType  entry_type;
    uint32      crl_serial;
    Date        last_crl;
    Date        this_crl;
    Date        next_crl;
    CRLEntry    entries<2^64-1>;

    SignerInfo   signer;
} UnsignedCRL;
```

```
struct {
    select(certificate_type){
        case ca:
        case psobu:
        case rsu:
            CertificateHash cert_hash;
        case obu:
```

```

        OBUCRLLinkage obu_link;
    }
} CRLEntry;

```

The `ca_id` field contains the CA for which this CRL is being issued.

The `crl_series` represents the CRL series that this CRL is for.

The `certificate_type` indicates the kind of certificates that this CRL is for.

The `crl_serial` is simply a counter that should increment by 1 for every issued CRL by that CRL issuer.

The `last_crl` and `this_crl` fields specify the time period that this CRL covers. Thus, any certificate that was revoked between `last_crl` and `this_crl` must appear on the CRL. Note that there is no requirement that the time windows represented by any two CRLs not overlap. This allows the issuance of delta CRLs or sliding window delta CRLs [45].

The `next_crl` value contains the time when the next CRL is expected to be issued. See Section 5.3.4.5.6 for discussion of the behavior of recipients with respect to CRLs.

`Entries` contains a list of the certificates that were revoked during this time period.

As usual, the `signer` field specifies the public key used to sign the CRL. There are two possibilities for the CRL signer:

1. It is the CA that originally issued the certificate.
2. It is a CRL signer authorized by the CA to issue CRLs.

In the latter case, the `signer` field must be of type `certificate` and the certificate must be of type `crlsigner`. That certificate must be directly signed by the original CA contained in the `signer_id`. The `responsible_series` field of the certificate must contain the value of the `crl_series` field of the CRL.

The `length` field of the `OrdinaryCRL` structure contains the length of the remaining structure (i.e. the combined length of the `unsigned_crl` and signature).

The `signature` field contains a digital signature over the encoded `unsigned_crl` value.

Partitioned CRLs

In order to keep down the size of individual CRLs, it may be desirable to maintain multiple CRL series for any given CA. Accordingly, a CA might break up the CRLs it issues into units of (say) 10000 certificates, each of which is issued on its own CRL series. Thus, a receiver would only need the relevant CRL for the certificate it was trying to verify rather than all CRLs for a given CA. The `crl_series` value is intended for this purpose. The number of CRL series and the partitioning of CRLs into a individual CRL series is a local matter for the CA, with one exception: OBUs must be randomly assigned to CRL series so that the CRL series in an OBU certificate does not leak information about the vehicle identity.

Non-Anonymous CRLs

In non-anonymous CRLs, used for all units besides OBUs, the certificates on a CRL are specified by `CertificateHash` .

```
opaque CertificateHash [10];
```

The CertificateHash must be the high order 10 bits of a SHA-1 digest of the revoked certificate.

Entries on a non-anonymous CRL must be sorted in increasing order of SHA-1 hash. Comparisons are done in big- endian format, treating the first byte of the SHA-1 hash as the most significant bit and each successive byte as less significant. This allows the receiving unit to do a binary search to determine if a certificate is on the CRL.

Anonymous CRLs

In the case where the revoked unit is an OBU, the CRLentry contains an OBUcRLLinkage value:

```
opaque          OBUcRLLinkage <1 .. 2^8-1>;
```

The OBUcRLLinkage simply contains the unit key k, described in Section 5.3.1.5.1.

Note that the process of verifying whether an OBU certificate is on a CRL is modestly computationally expensive: it requires precomputing the linkage values for all the current revoked units for the value of i in the certificate. In general we expect that units that need to verify OBU signatures will have an AES accelerator. Currently available cheap AES cores can perform on the order of 108 AES operations per second and could therefore check a million revoked OBUs in .01 s. Implementations may choose to cache known to be valid certificates once this check has been performed.

5.3.3 VSP Message Format

There are two kinds of messages that are transmitted over the air in the VSC system:

- Safety messages
- Updates

However, both kinds of message have a common format:

```
enum { signed(0), (255)} MessageType;

struct {
    uint8          protocol_version; /* 1 for this version */
    MessageType    type;
    opaque         message<2^16-1>;
} VSPMessage;
```

The `protocol_version` contains the current version of the protocol. The version described in this document is version 1, represented by the integer 1. There are no major or minor version numbers.

The type field contains the type of the message. This tells the receiving unit how to interpret the message body.

Currently the only defined types is signed (0) indicating a signed message. This leaves room for up to 254 other message types. All message types must be standardized before they may be used in real systems. Message types from 240 through 255 will not be assigned and are reserved for private usage in test environments. Production units must not generate messages with these types.

The message field is simply a block of bytes. These bytes are interpreted according to the type field. In particular, if the type field is signed then the message block is a `SignedMessage`, as specified in the next section.

5.3.4 Signed Message Format

Most of the messages in the VSP system are of type `SignedMessage`. The full PDU on the air thus looks something like the figure below:

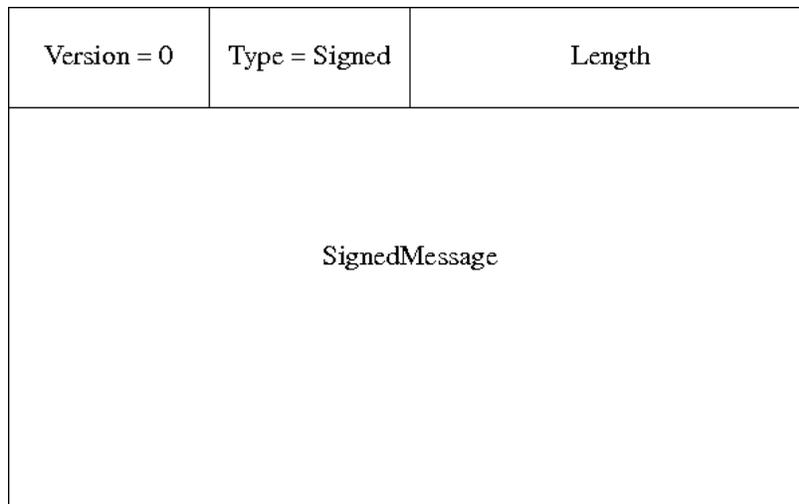


Figure 16: Subset difference broadcast tree

The specification for a `SignedMessage` is:

```

struct {
    ApplicationID    application;
    opaque          flags<0 .. 2^8-1>;
    opaque          application_data<1 .. 2^16-1>;
    opaque          message_id[2];
    Time            transmission_time;

```

```

        3DLocation    transmission_location;
        SignerInfo    signer;
    } ToBeSigned;

    struct {
        ToBeSigned    unsigned_message;
        Signature      signature;
    } SignedMessage;

    uint64            Time;

```

The application field contains the type of message being transmitted.

The flags field contains message-specific flags represented as a big-endian integer. A flag is set by ORing 2 flag it into the flags vector. The flags field should be the minimum necessary length. Thus, a vector where flags 2 and 5 was set would be the bytes 01 24. The only flag available is fragment (0) indicating that the message has been fragmented using the procedure specified in Section 5.3.5.4.

The application_data field contains the data for the application. This is not interpreted by the security protocol and is passed unchanged up to the application.

The message_id is a two-byte string that is unique for the transmission_time value.

The transmission_time field contains the time at which the message was generated in microseconds since the UNIX epoch (midnight, January 1, 1970).

The transmission_location field contains the location to which the message applies. For an OBU, this should be the current location of the unit as indicated by the GPS. An RSU, however, may "speak for" a physical unit that is not exactly in the same location as the antenna. For instance, a single RSU might broadcast messages for stop signs on four corners of an intersection. Such usages are explicitly permitted.

The signer value contains enough information to determine the keying material used to sign the message – though not necessarily the identity of the signer.

The signature block contains the digital signature itself.

Location Encoding

VSP uses lat/long/altitude coordinates to represent position, encoded in the Location structure:

```

    struct {
        opaque    latitude[5];
        opaque    longitude[5];
        uint16    altitude;
    }

```

```

} 3DLocation

struct {
    opaque    latitude[5];
    opaque    longitude[5];
} 2DLocation

```

`Latitude` and `longitude` are encoded as described in RFC 3825 [47], with a 6-bit resolution field and a 34-bit 2s complement fixed-point value. Implementations should use a resolution no less fine than 1 meter.

`Altitude` contains the an altitude position indicator in meters. All positions must use the WGS 84 [48] Datum.

Signer Info

The `SignerInfo` structure allows the recipient of a message to determine which keying material to use to authenticate the message:

```

struct {
    SignerIdentifierType  id_type;

    select (id_type) {
        case certificate:
            VSPCertificate certificate;
        case certificate_digest:
            opaque    digest[20];
    }
} SignerInfo;

enum { certificate(0), certificate_digest(1), (255)} SignerIdentifierType.

```

Currently, the `SignerInfo` may contain either a certificate or a message digest of a certificate. If the `id_type` field contains `certificate` (0) then the `SignerInfo` contains a certificate. If the `id_type` field contains `certificate_digest` (1) then the `SignerInfo` contains the `SignerID` corresponding to the relevant certificate.

Signature

The signature field contains the actual signature. The signature algorithm is uniquely determined by the type of the key used to generate the signature. There are no in-message indicators for signature algorithm type.

The signature is computed over the encoding of the entire `ToBeSigned` structure. Thus, for instance, if ECDSA is used as the signature algorithm the input message `M` to `Hash(M)` is the value of `unsigned_message`.

The signature value is simply a byte-string encoded in a signature algorithm-specific manner. Currently, the only algorithm so defined is ECDSA. Section 5.3.4.3.1 provides details on the use of ECDSA in VSP.

ECDSA Signatures

ECDSA signatures are performed as described in [46]. The input message is the encoded `unsigned_message` value. An ECDSA signature consists of two values `r` and `s`. These integers shall be converted into byte strings of the same length as the curve order `n` using the procedure of Section 4.3.1 of ANSI X9.62 [46]. The strings are padded with leading zeros to obtain the appropriate length. The strings are then concatenated to form the appropriate signature value:

```
struct {
    opaque[curve_order] r;
    opaque[curve_order] s;
} ECDSASignature;
```

Note that because the verifier knows the size of the curve, this structure can be unambiguously parsed despite the lack of length fields.

Transmission Processing

RSU and PSOBUS transmission processing is straightforward:

1. Generates a random message ID.
2. Get the current position and time values.
3. Encode the `unsigned_message` value.
4. Digitally sign the `unsigned_message` value.
5. Create and encode the `SignedMessage` value.

OBU transmission processing is similar but with one important difference: the message signature must be performed in a tamper-resistant hardware security module (HSM). The `message_id`, `transmission_time`, and `transmission_location` values must be generated inside the HSM. Thus, the tamper-resistant module must contain a real-time clock, GPS processor, and the cryptographic keying material. This ensures that the unit cannot forge messages with arbitrary positions or timestamps.

Reception Processing

When a unit receives a `SignedMessage` it must use the following process – or its equivalent – in processing it:

1. Decode the message.
2. Check that the `transmission_time` is within the acceptable time window. If not, discard the message.
3. Check that the `transmission_position` is within the acceptable position window. If not, discard the message.
4. Look up the message in the cache of recently received messages. If the message has already been received, discard it as a replay.
5. If the sender's certificate contains a scope restriction, verify that the `message_position` is within the geographic scope of every certificate in the sender's certification path. If not, discard the message.
6. Verify that the application field in the message is consistent with the scope restriction in the certificate.
7. Verify that the sender's certificate has not been revoked. If the sender's certificate has been revoked, discard the message.
8. Verify the sender's certificate. If the sender's certificate does not verify, discard the message.
9. Verify the signature on the message. If the signature does not verify, discard the message.
10. If all the previous tests verify, pass the message up to the application layer.

Note that this process is carefully designed to minimize the number of public key operations. Thus, checks 1-7 all can be performed with minimal computational overhead. Check 8 can be cached if the same certificate is seen multiple times. Thus, the only per-message computational overhead is seen in step 9. Implementations can use any equivalent process, however this algorithm is likely to be the most efficient.

Anti-Replay

Anti-replay in VSP takes advantage of the fact that each unit has an accurate clock. Thus, the clock in the sending unit (and, therefore, the timestamps in its messages) will be closely synchronized to that of the receiving unit. The receiving unit must use the following algorithm to prevent replays.

Each receiving unit maintains an anti-replay window of size $2r$; r should be at least 30 seconds. The receiver's idea of the current time is set to T . When a packet is received with timestamp t , the following procedure is followed:

1. If $t < T - r$ discard the packet as out of window.
2. If $t > T + r$ discard the packet as out of window.
3. If a copy of the packet is in the "already received" cache, discard the packet as a replay.
4. If the packet is accepted, add it to the already received cache and deliver it.

Every time that T advances, the already received cache should be emptied of all packets older than T-r. A ring buffer is a convenient data structure for this purpose; however, the exact choice of implementation is left up to the implementer.

Constructing the Certification Path

Certification path construction in VSP is an all-or-nothing proposition. Because messages must be processed in real time, implementations must either have the certificates on hand to verify a message or reject that message. Implementations must follow a procedure equivalent to the following algorithm, which is a form of the usual bottom up construction approach:

1. Set C equal to the current certificate and i=0
2. Set path[i]=C.
3. Set i=i+1.
4. If C is a trust anchor, the path is complete. Output path and i.
5. Parse C and extract the `signer_id` field
6. If there is a certificate in the local certificate cache such that the high order bits of `SHA1(S)` match the `signer_id`, call that certificate S. Otherwise, exit and output failure.
7. Set C=S.
8. Go to step 2.

The algorithm above delivers either an error or an unverified certification path of length i. Once the certificate is constructed, it must be verified, as described below. As noted before, we deliberately perform all the checks that can be performed with the unverified certification path before incurring the costs of signature verification.

Location Checks

Once the certification path is constructed, the recipient needs to verify that the location in the message is within the `GeographicScope` restrictions of the certification path. For each certificate in the path, the recipient must check that the location in the message is within that certificate's `GeographicScope`. Note that we do not require that the region permitted by each certificate be completely contained within the scope of the CA that signed that certificate. All that is necessary is that the location of the message be within the scopes of all the authorizing certificates in the certification path.

Usage Checks

The recipient must also check the usage restrictions on the certificates in the certification path. Each certificate except the end-entity certificate `path[0]` must be of type `ca`.

If the end-entity certificate is of type `rsu` or `psobu`, then the application field in the `UnsignedMessage` must be present in the application list in the certificate. If the end-entity certificate is of type `obu` the application field in the `UnsignedMessage` must be one of the applications that are permissible for OBUs. All application descriptions must describe whether they are permissible for OBUs or not. In addition, the recipient must check that the scope restrictions of the CAs in the path permit them to issue the certificates that they have issued. Thus, the application must also be permitted for each certificate in the path.

Check Certificate Cache

Recipients should maintain a cache of valid (both in terms of signature and CRL status). Implementations should check this cache prior to performing the remaining operations. A successful result allows bypassing of the remaining checks.

Certificate Revocation List Checking

Before a message is accepted, the recipient must check the appropriate Certificate Revocation List to determine whether the message was revoked at the time of message generation.

The appropriate CRL is defined as the one with a `ca_id` field that matches the certificate's `signer_id` field and a `crl_series` value that matches that in the certificate.

Once the correct CRL is identified, the recipient must then verify that the signing certificate does not appear on the CRL. For non-anonymous CRLs

, the recipient checks that the certificate hash does not appear on the CRL. For anonymous certificates, the recipient must verify that none of the linkage values in the CRL maps to the signing certificate.

As with any CRL-based system, the question arises of how to behave when the current CRL is not available. This is a particular problem in the VSP system because VSP CRLs are all push. Implementations have no ability to pull a CRL from the CA. Thus, an implementation that has been out of contact for a long period of time may get a message before getting the relevant CRL.

In order to balance the concerns of rejecting valid messages because we do not have a current CRL and of accepting invalid messages because we do not know that the signer is revoked, we recommend that recipients accept certificates for which a current CRL is not available provided that the most recent CRL is not too old. "Too old" is necessarily fuzzy, but we suggest no more than a month and no less than 50% of the time to the next CRL. For example if CRLs are issued on a monthly basis and the most recent CRL was issued on January 1. A recipient might accept certificates that were not revoked as of January 1st in the period February 1-February 14th, and perhaps through February 28th.

Checking Certificate Validity

Once all other checks have been performed, the recipient must check the validity of the signatures of each certificate in the certification path. Each certificate must be signed with the public key of the next higher-level certificate in the path.

5.3.5 System Updates

In order to maintain the VSC system, some data needs to be spread to most or all units in the system. At minimum, all units must contain a recent copy of the OBU CRL list. In addition, all units in a given geographic region need an up- to-date copy of the CRLs that correspond to RSO and PSOBUs certificates that are active in that area.

VSP has two mechanisms for widespread (whether global or local) propagation of update information:

1. Broadcast by RSUs
2. Flooding by OBUs

Updates enter the system by being broadcast by an RSU. They are then picked up by OBUs, which propagate them to other OBUs as necessary. A key element of this scheme is that updates can be broken up over multiple messages in such a way that once a critical number of messages are received the entire update can be reconstructed. Section 5.3.5.3 describes this process in detail.

Initial Update Injection

System updates are generated centrally and then propagated to RSUs via mechanisms that are outside of the scope of this protocol. For example, a transportation agency might generate a CRL for its RSUs and then deliver it to the RSUs it operates via wired Internet connections. Once the transmitting RSUs obtain a new update, they transmit it to units in their local area. Section 5.3.5.3 describes the correct transmission strategy.

Update Messages

There are two currently defined types of update, CRLs (application type `crl_transmit`) and certificates (application type `cert_transmit`). In both cases, the `application_data` field of the `signed_message` is set directly to the value of the encoded CRL or certificate. If the CRL or certificate is too large to fit in a single message, it must be fragmented, as described below.

Reliable Broadcast of Updates

In general, system updates will be far too large to fit in a single VSP Message structure. Thus, they must be split over multiple messages. In order to do this efficiently without requiring an acknowledgement channel, we use expandable forward error correction codes, as described in RFC 3450-3453 [49]. These codes allow split a message into pieces such that any subset of those pieces of combined size approximately that of the original message can be used to reconstruct the original message.

A sender starts with a message `M`. It then generates and transmits a series of symbols using the following algorithm:

1. Generate a random `symbol_id`.
2. Compute the corresponding symbol using `Ssymbol_id = FEC_encode(symbol_id,M)`.
3. Transmit the pair (`symbol_id,Ssymbol_id`).

Because the `symbol_id` is randomly generated, the sender can be stateless. As long as the `symbol_id` space is relatively large (e.g., 32 bits), the probability of symbol repetition is low. As many output symbols can be generated as the number of possible `symbol_ids`. In VSP, source symbols are transmitted using the following structure:

```
struct {
    opaque          update_digest[20];
    uint32         required_symbols;
    uint8          fec_scheme;
    uint32         source_block;
    uint32         symbol_id;
```

```
opaque          symbol<1 .. 2^16-1>;
} EncodedSymbol;
```

`EncodedSymbol` structures are constructed using the procedure described in RFC 3450.

`update_digest` is a SHA-1 hash of the original source message, which allows recipients to disambiguate different fragments for reassembly.

`required_symbols` contains the expected number of symbols which will be required for reconstruction. This allows the recipient to know when to start reconstructing.

Transmitting Update Fragments

A transmitting unit transmits an update as a stream of `EncodedSymbol` structures. In order to prevent denial of service attacks on individual update messages, each symbol must be signed. The symbols are transmitted as `SignedMessage` objects with the appropriate application ID, either `cert` or `crl`. Before attempting to reconstruct a message, the recipient should verify the fragment certificates. This does not ensure validity of the reconstructed message but merely ensures that the fragments were not tampered with, thus avoiding reconstruction errors. The recipient must still verify the signature on the reconstructed object (e.g., certificate or CRL).

5.4 Enrollment Procedures

The first thing that a unit must do in order to transmit VSP messages is to enroll and get certified keying material. In the case of CAs, RSUs and PSOBUs, this is a fairly straightforward procedure, described in Section 5.4.1. Enrollment for OBUs is complicated by the requirement for privacy and is described in Section 5.4.2.

5.4.1 CAs, RSUs, and PSOBUs

Because CAs, RSUs and PSOBUs are authenticated via ordinary PKI-style mechanisms, enrollment of these units can be done in the conventional way. To a first order, this is a manual process: the enrolling party (the future certificate holder) presents the CA with:

Its public key

1. Its desired authorization information (scope)
2. Documentation indicating that it is entitled to the desired certificate.

The CA then decides whether to issue a certificate or not based on its internal policies. If the policy checks succeed the CA returns a certificate to the unit.

Although these processes are inherently somewhat manual, experience with PKI systems has shown that it is helpful to have standard formats for certificate signing requests (CSRs) and delivery of the return certificates. We describe those here.

Certificate Signing Requests

In order to get a certificate signed, the enrollee first generates an asymmetric key pair and uses it to construct a `CertificateSigningRequest` object.

```
struct {
    uint8          csr_version;
```

```

    SubjectType    type;

    CertSpecificData type_specific_data<1 .. 2^16-1>.
    PublicKey      public_key;
} UnsignedCSR;

struct {
    UnsignedCSR    unsigned_csr;
    Signature      signature;
} CertificateSigningRequest;

```

The `csr_version` field should be set to 1 for this specification.

The `subject_type` should specify the desired type of certificate. Note that the value must not be `obu`.

The `type_specific_data` field should specify the desired scope. Note that this is rendered as a variable length value. This enables the enrollee to specify no scope value (by using a zero length) and let the CA insert a correct scope. Only one `CertSpecificData` structure should appear in this field.

The `public_key` field contains the public key of the enrollee.

The `signature` field contains a signature over the encoded `unsigned_csr` value using the private component corresponding to the `public_key` value.

Trust Anchor Certificates

In VSC, trust anchors are represented as self-signed certificates. These certificates are precisely like ordinary CA certificates except that they have a `signer_id` that consists entirely of zeros.

Certificate Issuance

Once a CA receives a CSR, it must compare it with the associated documentation (delivered in an unspecified manner) and decide whether the certificate can be issued according to the CA policies (also a local decision). If the certificate can be issued, the CA then formats the appropriate `UnsignedCertificate` structure and signs it to generate a `Certificate` value. Note that the CA is not bound by the enrollee's suggestion of subject type or scope. It may issue a certificate of any type or scope of its choosing. However, a CA should not generate certificates which will not result in verifiable certificate chains, e.g. by listing geographic scopes which are not within the CA's own scope.

Once the `Certificate` is created, the CA returns it in a `CertificateResponse` structure.

```

struct {
    Certificate     certificate_path<1 .. 2^32-1>;

```

```
        OrdinaryCRL    crl_path<1 .. 2^32-1>;  
    } CertificateResponse;
```

The `certificate_path` contains the certificate path of the new certificate. This path is in order, with the most local certificate (the newly issued one) being first and each successive certificate signing the one before it. The path should be complete with the final certificate being a trust anchor. However, some implementations may choose to deliver less complete paths for space reasons.

The `crl_path` contains the CRLs necessary to validate the certificate. At minimum, it must contain the most recent version of the CRL series on which the issued certificate would appear if it were revoked. In addition, CAs should include CRLs corresponding to other CAs in the chain. These CRLs are not ordered.

5.4.2 OBUs Certificate Initialization

OBUs should be initialized by their manufacturers. This initialization must happen in such a way that no identifying information is stored linking any given OBU certificate to the identity (serial number, physical identity, etc. of the OBU).

The VSC team anticipates that the OBU CA and the OBU to be initialized will be physically collocated. When the operator of the CA authorizes the initialization of the OBU, the OBU generates a series of `CertificateSigningRequests` and provides them to the CA. The OBU also provides the encrypted `h[i],B[i]` pairs to the CA. Note that the escrow authority may not (and probably should not) be the same as the CA. The CA then signs the certificates and returns the `CertificateResponse` values. At this point the OBU is initialized.

5.5 Suggested Operating Procedures

This section provides identification of some of the security architecture operating requirements that will arise, and suggests a solution.

5.5.1 OBU Operating Requirements

Because OBUs are allowed to send so many different kinds of messages from so many different locations, the OBU must be designed to restrict the class of messages that can be sent.

The OBU signing keys must be embedded in a tamper-resistant Hardware Security Module (HSM). This HSM must be compliant with FIPS 140-2 level 3. The HSM must be designed not to release the OBU signing keys from the module. In addition, it must not be usable for signing arbitrary messages.

All messages signed by the HSM must be wrapped in a `SignedMessage` structure. The HSM must populate the `message_id`, `transmission_time`, and `transmission_location` fields. The `transmission_time` and `transmission_location` fields must be populated with data received respectively from a clock and GPS unit, which are housed within a FIPS 140-2 level 3 module. It is also recommended that the clock and GPS unit be housed within the same module as the signing module. However, if they are housed within a separate unit, then communications between the two modules must be authenticated with an algorithm that provides at least 100 bits of security and measures must be taken to ensure timing synchronization between the two modules.

The clock must be periodically updated from the GPS unit in order to correct for clock slip. However, because the GPS unit gets its input from radio signals outside the tamper boundary, mechanisms must be used to isolate the system from GPS spoofing. The clock must be calibrated

for maximum slip values and must not allow for corrections beyond those values. In addition, "backward" corrections must be performed by slow-ticking rather than by rolling time backward. In addition, the system should enforce physical plausibility rules, such as rejecting speeds in excess of the maximum speed of the vehicle, impossible altitudes, etc.

5.5.2 OBU CAs and Escrow Authorities

In order to preserve privacy, it is critical that no records be kept of the connection between any individual OBU and the certificates that it was issued. This requires certain procedures from CAs and escrow authorities.

5.5.3 CA's Procedures

For signature algorithms that are compatible with blind signatures, certificates should be signed using a blind signature protocol. However, not all algorithms are compatible with blind signatures. In particular, ECDSA-- which is the only currently defined algorithm--is not. An alternate mechanism is therefore described that should be used in such cases.

The actual CA signing unit should be embedded in a FIPS 140-2 level 4 tamperproof module. The OBU should be physically connected to the CA signing unit (e.g. via FireWire or USB) and should establish a cryptographically secure connection using TLS [6] (the CA's TLS certificate should be wired into the OBU at manufacture). All communications should take place over this channel. Once the certificates have been issued the CA should output the encrypted $h[i], B[i]$ pairs and then delete all records of the transaction.

No OBU-specific identifying information should be passed over this channel. Instead, the authorization for this process should be performed by placing the CA in "active" mode. Once the CA is in active mode, it will sign certificates for some number of OBUs without knowing their identities. In order to protect the CA from theft and misuse, the ability to activate the CA should be controlled via conventional cryptographic token and secret sharing technology.

Escrow Authorities

As described in Section 5.3.1.5.1, the ability to link certificates from the same OBU is reserved to Escrow Authorities. In order to preserve privacy, Escrow Authorities should not be the same as the OBU CAs.

Escrow Authorities must keep linkage information under strict physical security and make best commercial efforts to retain confidentiality of said records. Access to the escrowed $h[i], B[i]$ pairs should be controlled by a FIPS 140-2 level 4 device. This device should restrict the number of linkage events to a limited number per day.

5.6 OBU Privacy Parameters

This section of the document addresses OBU privacy parameters.

5.6.1 Choice of Work Factor for OBU Blinding

Section 5.3.1.5.1 describes a mechanism for increasing the cost of de-escrowing OBU certificates. The rationale here is that OBU revocation will happen relatively rarely and therefore need not be inexpensive. Making the cost of linking OBU certificates relatively high reduces the probability that VSC can be used as a generic tracking system for a large number of OBUs. Thus, w should be chosen so that it is possible to de-escrow OBU certificates when necessary for revocation but sufficiently expensive to make it impractical to do so on a mass basis.

The correct metric for choosing w is cost. It should be sufficiently high that casual de-escrowing is expensive but sufficiently low that it is practical to de-escrow a key. In our constraints analysis,

we estimated that \$5,000 would be the target investigative cost for incidents caused by a malicious OBU. Therefore, we recommend that w be chosen so that the cost of revoking an OBU once its certificate is identified be of the same order of magnitude, between \$1,000 and \$5,000. OBU manufacturers should periodically adjust the value of w upward to keep pace with the increasing speed and decreasing cost of hardware.

5.6.2 OBU Certificates

Giving each OBU a large number of certificates provides unlinkability against ordinary attackers. An attractive feature of the VSP protocol design is that the size of protocol messages is independent of the number of certificates per OBU. Thus, the only constraints on the number of certificates per OBU are the cost of initial OBU initialization and OBU memory.

We recommend that at minimum each OBU be issued 5,000 certificates. In order to minimize information leakage, signing keys should be randomly selected and used for a short time period and then discarded. Implementations may use the following algorithm, which produces a key half-life of h seconds or m miles, for key selection, with a uniform probability that a key will be discarded in any time period.

1. Select a random number r uniformly from the set $[1,n]$
2. Set the current key to $K[r]$. Set t_0 to the current time in seconds.
3. As each second passes, with probability $[\log(2)]/h * e^{\log(.5/h)}$ discard the key and go to step 1.
4. As each mile passes, with probability $[\log(2)]/m * e^{\log(.5/m)}$ discard the key and go to step 1.

Because the probability that a key will be discarded is uniform, this leaks the minimal amount of information.

We recommend values of $h=600$, $m=10$. With these values, a vehicle moving at 60 mph will have a key half-life of 5 minutes. As a special case, we recommend not changing keys when the vehicle is not in motion. This prevents an attacker from observing large number of keys from parked running vehicles.

5.7 Opportunities for Optimization

Compactness of encoded protocol data units is an important consideration for VSC. VSP has, therefore, been designed to be fairly parsimonious with respect to PDU size. However, in the interest of maximizing protocol and design flexibility we have deliberately chosen not to make a number of small optimizations that would individually save a byte or two. It may be discovered that these optimizations pay off in the future as the protocol becomes more refined. Some such optimizations are listed here.

5.7.1 Combined Algorithm Identifiers

Currently VSP uses separate algorithms and parameters. Thus, ECDSA has a parameter specifying the group being used. This preserves the maximum amount of flexibility because it allows an arbitrary amount of information to be carried in algorithm parameters. However, it also consumes additional space and is unnecessary if there are only a small number of algorithm/parameter pairs. Identifying these pairs by a single integer would save a byte or two per certificate.

5.7.2 Length of Fields

VSP makes use of variable-length fields in a number of locations. These fields are denoted by use of the notation `<YYY - XXX>` where `XXX` is the maximum length of the field. When such fields are encoded, they are encoded by first encoding the length and then the value. The length is encoded as a fixed-width value of 1-byte multiples. Thus, in order to encode a field that might be 256 bytes long, we need a 2-byte length field – the same size length field as would be required for a value with a maximum length of 65535 bytes. Thus, whenever we wish to reserve a length larger than that which would fit in `x` bytes, we instead use the largest length that would fit in `x+1` bytes. It is not expected that fields of that size will ever appear.

There are two potential optimizations here:

1. Restrict maximum lengths more tightly in the protocol. E.g., use opaque field `<1 .. 500>`.
2. Use smaller maximum length values, thus saving length bytes.

The first option only prevents programmers from using large data sizes, but doesn't provide an immediate savings. The second option provides a single byte saving for every field where the field is represented.

5.8 Summary of Syntax

This section provides a summary of the syntax discussed heretofore:

```
struct {
    uint8      certificate_version;
    SubjectType subject_type;
    CertSpecificData type_specific_data;
    Date       expiration;
    CRLSeries  crl_series;
    SignerID   signer_id;
    PublicKey  public_key;
} UnsignedCertificate.
```

```
opaque[8]    SignerID;
uint16       CRLSeries;
```

```
struct {
    select(subject_type){
        case ca:
            CAScope  scope;
        case rsu:
            RSUScope scope;
        case psobu:
```

```

        PSOBUScope scope;
    case obu:
        LinkageData linkage;
    case crlsigner:
        CRLSeries responsible_series<2^16-1>;
    }
} CertSpecificData;

enum { ca(0), rsu(1), psobu(2), obu(3), crlsigner(4), (255) } SubjectType;

uint16 Date;

struct {
    SignatureAlgorithm algorithm;

    select(algorithm){
        case ecdsa:
            ECDSAKey key;
        }
} PublicKey;

struct {
    opaque point<1..2^8-1>;
} ECPPoint;

enum { ecdsa(0), (255) } SignatureAlgorithm;

struct {
    uint16 length;
    UnsignedCertificate unsigned_certificate;
    Signature signature;
} VSPCertificate

opaque Signature<1 .. 2^16-1>;

struct {
    NamedCurve curve;
    ECPPoint point;
} ECDSAKey;

enum {

```

```

K-163(1),
B-163(2),
K-233(3),
B-233(4),
reserved (5..239),
  private (240..255)
} NamedCurve;

```

```

struct {
  opaque point <1..2^8-1>;
} ECPoint;

```

```

struct {
  ApplicationID      applications<0 .. 2^16-1>;
  SubjectType       unit_types<1 .. 2^8-1>;
  GeographicRegion  region;
} CAscope.

```

```

struct {
  ApplicationType  type;
  ApplicationSubtype subtype;
} ApplicationID;

```

```

enum {cert_transmit(0), crl_transmit(1), (2^16-1)} ApplicationType;
opaque ApplicationSubtype<0 .. 255>;
enum {fragment(0), (2048)} MessageFlags;

```

```

struct {
  RegionType region_type;

  select(region_type){
    case polygon:
      PolygonalRegion polygonal_region;
    case circle:
      CircularRegion circular_region;
    case rectangle:
      RectangularRegion rectangular_region<2^16-1>;
  }
}

```

```

    }
}

enum { polygon(0), circle(1), rectangle(2), (255) } RegionType;

```

```

PolygonalRegion    2DLocation<2^16-1>;

```

```

struct {
    2DLocation    center;
    uint16    radius;
} CircularRegion;

```

```

struct {
    2DLocation    upper_left;
    2DLocation    lower_right;
} RectangularRegion;

```

```

struct {
    GeographicRegion    region;
    ApplicationID    applications<0 .. 2^16-1>;
} RSUScope;

```

```

struct {
    GeographicRegion    region;
    ApplicationID    applications<0 .. 2^16-1>;
} RSUScope;

```

```

struct {
    GeographicRegion    region;
    ApplicationID    applications<0 .. 2^16-1>;
} PSOBUScope;

```

```

struct {
    opaque    enc_w<2^8-1>;
    uint16    i_value;
}

```

```

} LinkageData;

struct {
    uint32    length;
    UnsignedCRL  unsignedCRL;
    Signature  signature;
} OrdinaryCRL;

struct {
    CRLSeries    crl_series;
    SignerID     certificate_type;
    SubjectType  entry_type;
    uint32       crl_serial;
    Date         last_crl;
    Date         this_crl;
    Date         next_crl;
    CRLEntry     entries<2^64-1>;

    SignerInfo   signer;
} UnsignedCRL;

struct {
    select(certificate_type){
        case ca:
        case psobu:
        case rsu:
            CertificateHash cert_hash;
        case obu:
            OBUCRLLinkage obu_link;
    }
} CRLEntry;

opaque CertificateHash[10];

opaque          OBUCRLLinkage <1 .. 2^8-1>;

enum { signed(0), (255)} MessageType;

```

```

struct {
    uint8          protocol_version; /* 1 for this version */
    MessageType    type;
    opaque         message<2^16-1>;
} VSPMessage;

```

```

struct {
    ApplicationID  application;
    opaque        flags<0 .. 2^8-1>;
    opaque        application_data<1 .. 2^16-1>;
    opaque        message_id[2];
    Time          transmission_time;
    3DLocation    transmission_location;
    SignerInfo    signer;
} ToBeSigned;

```

```

struct {
    ToBeSigned    unsigned_message;
    Signature     signature;
} SignedMessage;

```

```

uint64          Time;

```

```

struct {
    opaque        latitude[5];
    opaque        longitude[5];
    uint16        altitude;
} 3DLocation

```

```

struct {
    opaque        latitude[5];
    opaque        longitude[5];
} 2DLocation

```

```

struct {
    SignerIdentifierType  id_type;
}

```

```

    select (id_type) {
        case certificate:
            VSPCertificate certificate;
        case certificate_digest:
            opaque    digest[20];
    }
} SignerInfo;

enum { certificate(0), certificate_digest(1), (255) } SignerIdentifierType.

```

```

struct {
    opaque[curve_order] r;
    opaque[curve_order] s;
} ECDSASignature;

```

```

struct {
    opaque    update_digest[20];
    uint32    required_symbols;
    uint8     fec_scheme;
    uint32    source_block;
    uint32    symbol_id;
    opaque    symbol<1 .. 2^16-1>;
} EncodedSymbol;

```

```

struct {
    uint8     csr_version;
    SubjectType    type;

    CertSpecificData type_specific_data<1 .. 2^16-1>.
    PublicKey    public_key;
} UnsignedCSR;

```

```

struct {
    UnsignedCSR    unsigned_csr;
    Signature    signature;
} CertificateSigningRequest;

```

```
struct {  
    Certificate    certificate_path<1 .. 2^32-1>;  
    OrdinaryCRL   crl_path<1 .. 2^32-1>;  
} CertificateResponse;
```

6 References

1. CAMP VSC Consortium, *Vehicle Safety Communications Project, Task 3 Final Report* (May, 2004).
2. Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280 (April 2002).
3. NIST, "Security Requirements for Cryptographic Modules," FIPS PUB 140-2 (2002).
4. Rescorla, E., Korver, B., and IAB, "Guidelines for Writing RFC Text on Security Considerations," RFC 3552 (May 2003).
5. Bellardo, J., and Savage, S., "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions," *Proceedings of the 12th USENIX Security Symposium* (August 2003).
6. Frederick, et al., "Cellular Telephone Fraud Anti-Fraud System," *US Patent 5,448,760* (September 5, 1995).
7. Chaum, D., and van Heijst, E., "Group Signatures," *EUROCRYPT 91*, pp. 256-265, Springer-Verlag (1991).
8. Tsudik, G., and Xu, S., *Accumulating Composites and Improved Group Signing*.
9. Viega, J., and McGraw, G., *Building Secure Software*, Addison-Wesley (2002).
10. Chesson, G., Personal communication.
11. Rescorla, E., "Security Holes... Who cares?," *Proceedings of the 13th USENIX Security Conference* (2003).
12. Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280 (April 2002).
13. Housley, R., "Cryptographic Message Syntax," RFC 3369 (August 2002).
14. Nagle, J., "Congestion Control in IP/TCP Internetworks," RFC 0896 (Jan 1984).
15. Kent, S., Atkinson, R., and RFC 2406, *IP Encapsulating Security Payload (ESP)* (November 1998).
16. Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP," RFC 2560 (June 1999).
17. Scott, M., *Multiprecision Integer and Rational Arithmetic C/C++ Library*.
<http://indigo.ie/~mscott/>
18. Rivest, R.L., Shamir, A., and Adelman, L.M., "On Digital Signatures and Public Key Cryptosystems," Technical Report, MIT/LCS/TR-212, MIT Laboratory for Computer Science (January 1979).
19. Boneh, D., Gentry, C., Lynn, B., and Shacham, H., "A Survey of Two Signature Aggregation Techniques," *RSA CryptoBytes*, 6, 2 (2003).
20. Rivest, R., Shamir, A., and Adleman, L.M., "Cryptographic communications system and method," US Patent 4405829 (September 1983).

21. National Institute of Standards and Technology (NIST), and Digital Signature Standard, FIPS PUB 186-2, U.S. Department of Commerce (2000).
22. Kravitz, D., "Digital signature algorithm," U.S. Patent 5,231,668 (July 1993).
23. Schnorr, K., "Method for Identifying Subscribers and for Generating and Verifying Electronic Signatures in a Data Exchange System," US Patent 4995082 (Feb 1991).
24. Certicom, *Possible patents pending Claim on ECDSA*.
<http://www.ietf.org/ietf/IPR/CERTICOM-ECDSA>
25. Pintsov, L.A., and Vanstone, S.A., "Postal Revenue Collection in the Digital Age," *Lecture Notes in Computer Science*.
26. Boneh, D., Lynn, B., and Shacham, H., "Short Signatures from the Weil Pairing," *Proceedings of Asiacrypt 2001* (2001).
27. Naor, D., Naor, M., and Lotspiech, L., "Revocation and Tracing Schemes for Stateless Receivers," *Lecture Notes in Computer Science* (2001).
28. Postel, J., "Transmission Control Protocol," RFC 793 (September 1991).
29. Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols," *ACM Computer Communication Review* (1997).
30. Sasson, Y., Cavin, D., and Schiper, A., "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," *Proceedings of IEEE Wireless Communications and Networking Conference*.
31. Cartigny J., Simplot D., and Carle, J., *Stochastic flooding broadcast protocols in mobile wireless networks*.
32. Chaum, D., "Blind Signature Systems," *Advances in Cryptology, Crypto '83* (1983).
33. Camenisch, J.L., Piveteau, J., and Stadler, M.A., "Blind Signatures Based on the Discrete Logarithm Problem," *Lecture Notes in Computer Science* (1995).
34. Rabin, M.O., "Digital Signatures," *Foundations of Secure Communications* (1987).
35. Garay, J., Staddon, J., and Wool, A., "Long-Lived Broadcast Encryption," *Lecture Notes in Computer Science* (2000).
36. Jiang, D., "Security Consideration for Vehicle Safety Communications," *ASTM DSRC standardization meeting* (2002).
37. Chaum, D., and van Heyst, E., "Group Signatures," *Advances in Cryptology — EUROCRYPT 91* (1991).
38. Ateniese, G., Camenisch, J., Joye, M., and Tsudik, G., "A Practical and Provably Secure Coalition-Resistant Group Signature Scheme," *Lecture Notes in Computer Science* (2000).
39. Menezes, A., *Elliptic Curve Public Key Cryptosystems*, Kluwer (1993).
40. Lenstra, A.K., and Verheul, E.R., "Selecting Cryptographic Key Sizes," *Journal of Cryptology*, 14, 4, pp. 255-293.
41. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S. "Analysis of an Electronic Voting System", *IEEE Symposium on Security and Privacy 2004*, May 2004.
42. Upton, J., "Gadget may wreak traffic havoc", *The Detroit News*, October 26, 2003.
Klein-Berndt, J. "NIST Summary of Simulation Results", September 30, 2004.

43. Dierks, T., and Allen, C., "The TLS Protocol Version 1.0," RFC 2246 (January 1999)
44. Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS).," RFC 3565 (July 2003).
45. Cooper, D. A., "A More Efficient Use of Delta-CRLs, "Symposium on Security and Privacy, pp. 190-202.
46. ANSI, "Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", " ANSI X9.62 (1998).
47. RFC 3825: Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information. J. Polk, J. Schnizlein, M. Linsner. July 2004. (Format: TXT=31715 bytes) (Status: PROPOSED STANDARD)
48. World Geodetic System 1984 (WGS 84), MIL-STD-2401, <http://www.wgs84.com/>
49. 3450 Asynchronous Layered Coding (ALC) Protocol Instantiation. M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, J. Crowcroft. December 2002. (Format: TXT=86022 bytes) (Status: EXPERIMENTAL)

- 3451 Layered Coding Transport (LCT) Building Block. M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, J. Crowcroft. December 2002. (Format: TXT=72594 bytes) (Status: EXPERIMENTAL)

- 3452 Forward Error Correction (FEC) Building Block. M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, J. Crowcroft. December 2002. (Format: TXT=38368 bytes) (Status: EXPERIMENTAL)

- 3453 The Use of Forward Error Correction (FEC) in Reliable Multicast. M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, J. Crowcroft. December 2002. (Format: TXT=46853 bytes) (Status: INFORMATIONAL)
50. X9.63: American National Standard for Financial Services. ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography. November 2001.

7 Exhibit A: Analysis of ECDSA Performance

Only "even" elliptic curves (EC) will be considered here (i.e., EC over GF(2^m)). "Odd" EC in general would be fairly comparable – but probably somewhat worse – in speed and size. The silicon industry in general has not seen many chips containing EC, so our estimates are somewhat theoretical in nature, as opposed to the modular math units discussed above for RSA and DSA, where many vendors have produced chips in recent years. However, our estimates here are quite encouraging.

The "straightforward" algorithms used to implement EC are explained in the book *Elliptic Curve Public Key Cryptosystems* by Alfred Menezes [39]. While it is generally believed possible to implement such EC functions without violating any patents, Certicom is almost guaranteed to disagree, and it may be prudent to obtain a license from Certicom, which fortunately is in the licensing business.

The basic step in an ECDSA signature is a point multiply, while the verify uses two such multiplies. Thus, verification is basically half the speed of signatures. All other operations involved are quite short, from a time perspective. Using the Menezes straightforward approach (page 22), each point doubling requires 1 field inversion, 4 field additions, 2 squares, and 3 multiplications. A point addition requires Over GF(2^m), a field inversion can be performed in about 2^m clocks using Euclid's algorithm, and a square and a multiplication each take about $m/2$ clocks, assuming we process two multiplier bits per clock. We use no normal bases, which would speed up the squares, since it slows down and complicates all the other steps rather significantly. If we assume that each field operation has an overhead of H clocks (where H is small, say 4), then a point doubling takes about

$$(2m + H) + 4H + 2(m/2 + H) + 3(m/2 + H) = 4.5m + 10H \text{ clocks}$$

and a point addition takes about

$$(2m + H) + 8H + (m/2 + H) + 2(m/2 + H) = 3.5m + 12H \text{ clocks}$$

For a point multiplication, assuming a random multiplier (i.e., with Hamming weight $m/2$), there are m point doublings and about $m/2$ point additions, for a total of

$$6.25m^2 + 16Hm \text{ clocks}$$

Using $m = 163$, which is one of the NIST standard curves for ECDSA, and assuming a frequency of 200 MHz, we obtain a point multiplication time of about 0.8 msec, which easily achieves our signature and verification latency. By building two of these units, we can achieve a throughput of over 2000 ECDSA verifications/sec. The silicon area of each unit should be under 40 Kgates, so the whole core would occupy roughly 75 Kgates, with an incremental cost under \$0.30 and a rough power consumption of under 0.2 Watts.

Observe that, if these EC numbers do not quite achieve the speed required (e.g., if these theoretical estimates are slightly off), then there are several avenues available for optimization. First, the frequency of operation can probably be higher than 200 MHz fairly easily, as GF(2^m) field operations never involve carry propagation. Second, it is possible to use somewhat more gates to increase the speed of squares and multiplies, by processing more multiplier bits per cycle. Third, standard windowing techniques for exponentiation could be used to minimize the number of point additions required. Finally, Certicom has some nice pending patents on optimizations that could be licensed to minimize the number of field operations required, basically by removing

the field inversions, without adding significantly to the silicon area. In any case, it seems quite clear that the performance goals for ECDSA can be met at a very reasonable silicon cost.

8 Exhibit B: Choice of Key Size

Although the protocols should be designed to work with a variety of key sizes, in this section we provide guidance for appropriate key sizes. Cryptographic key size is generally a choice between performance (both computation time and message size) and security. The following table (from [32]) shows the approximate strengths and sizes for a variety of ECDSA key sizes (all values in bits).

Key Size	Strength	Signature Size
130	70	130
148	75	148
165	80	165
185	85	185
205	90	205
222	95	222
240	100	240

Table 5: ECDSA Key Strengths and Sizes

We recommend that RSU and OBU keys be chosen at the 80-bit security level. Breaking such keys is well outside current capabilities and is surely far more expensive than simply violating the tamper seal on an OBU or RSU. Compromise of such keys can be dealt with by ordinary revocation mechanisms.

We recommend that OBU CA keys be chosen to be stronger, because failure of such a key is catastrophic. Such keys should be chosen at at least the 90-bit security level. The root RSU CA key should be chosen to be at an equivalent security level. Intermediate and local CA keys can be weaker, especially if they are of limited scope or the keys themselves are of limited lifetime.

9 Exhibit C: The Subset Difference Algorithm

Subset difference is an advanced broadcast encryption scheme with superior properties to the subset cover scheme. As with subset cover, we arrange the nodes in a binary tree. The notation we use is that for any node N_s , the left child is called $N_{s,l}$ and the right child is called $N_{s,r}$. A typical tree is shown in the figure below.

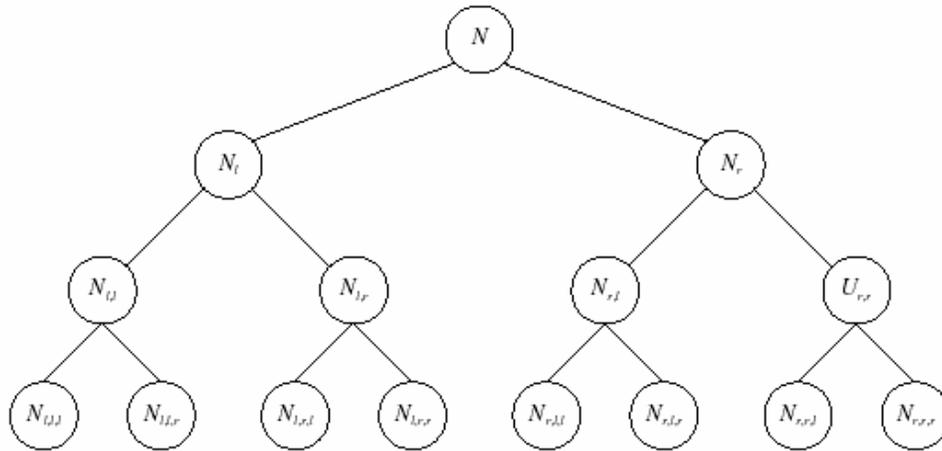


Figure 17: Subset Difference Broadcast Tree

Key assignment in subset difference is different from that of subset cover. In subset cover, a node gets the keys on the path from it to the root. In subset difference it gets the keys that are *off* the path to the root as well as for the children of those nodes. For instance, Figure 18 shows the nodes corresponding to the keys which node $N_{ll,r}$ has as shaded. Thus, somewhat counter-intuitively, node $N_{ll,r}$ receives the keys corresponding to nodes $N_{ll,l}$, $N_{ll,r}$ and N_r and their children, namely: $K_{ll,l}$, $K_{ll,r}$, $K_{l,r,l}$, $K_{l,r,r}$, $K_{r,l}$, $K_{r,l,l}$, $K_{r,l,r}$, $K_{r,r,l}$, and $K_{r,r,r}$.

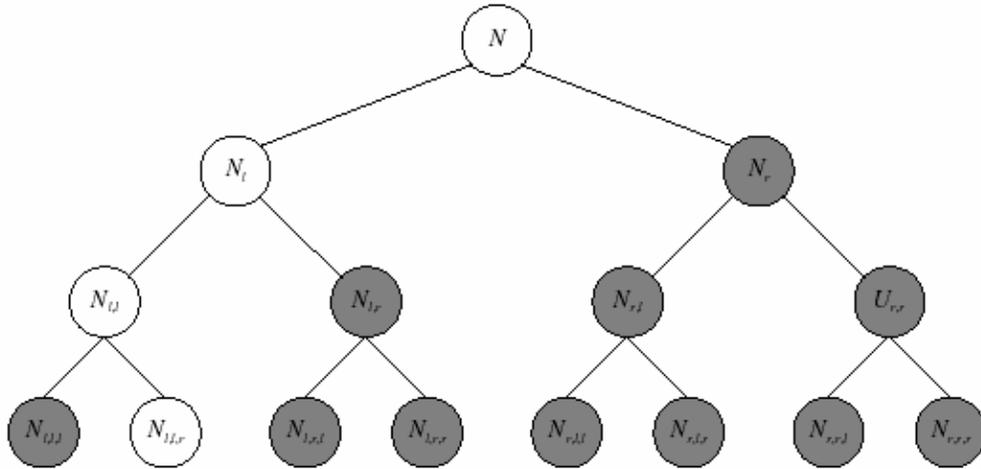


Figure 18: Keys Assigned to Node $N_{l,l,r}$

Encryption in subset difference is more complicated than in subset cover. The idea is to encrypt to subtrees with cutouts. So, for instance, we might encrypt to "every child of N except $N_{l,r}$ ". We call this subset $S(N, N_{l,r})$. Now, it should be clear that the only leaf node key that $N_{l,r}$ *doesn't* have is $K_{l,r}$. Moreover, it is the only node that doesn't have that key! So, if we want to encrypt to $S(N, N_{l,r})$, we counter-intuitively encrypt under key $K_{l,r}$. The nice thing about this scheme is that the ciphertext is much smaller. In order to revoke $N_{l,r}$ in the subset cover scheme we would have to encrypt under three keys (K_r , $K_{l,r}$, and $K_{l,r,r}$). Here we need only encrypt under one key.

The problem here is that each leaf node is issued a ridiculous number of keys, eleven in this very simple system of eight users. However, this can be fixed with a little clever design. Instead of making each key independent, we use hashes to derive the keys for each child node from those of the parent node. Thus, instead of having the key for every other leaf node, each unit just has the keys of all the "off-nodes" in the path to the root and can derive the remaining keys by successive hashing. Thus, the number of keys stored at the end-node is commensurate with that required for subset cover.

10 Exhibit D: Table of Acronyms

ACJT	Ateniese, Camenisch, Joye, Tsudik (group signature algorithm)
BLS	Boneh Lynn Shacham (public key signature algorithm)
CA	Certificate Authority
CRL	Certificate Revocation List
DOT	Department of Transportation
DSRC	Dedicated Short Range Communications
DSA	Digital Signature Algorithm (public key signature algorithm)
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
MAC	802.11 Media Access Control layer (also Message Authentication Code but not in this document--see MIC)
MIC	Message Integrity Check
OBU	On-Board Unit
OEM	Original Equipment Manufacturer
PHY	802.11 Physical layer
PKI	Public Key Infrastructure
PSOBU	Public Safety On-Board Unit
RSA	Rivest Shamir Adelman (public key cryptosystem)
RSU	Road-Side Unit
VIN	Vehicle Identification Number
VSC	Vehicle Safety Communications
VSCC	Vehicle Safety Communications Consortium