Chapter 1

INTRODUCTION

Digital broadcast technology is hot. A casual glance at a newspaper reveals it: articles describe the impact of streaming media, questions of intellectual property being broadcast, new applications that rely on broadcast, discussions of privacy concerns with broadcast, etc. We are invited to electronically join broadcast meetings over the Internet, we can choose to listen to famous or obscure streaming radio stations from most corners of the globe, and we are promised a future where we will receive high-definition digital TV.

Clearly, broadcasting is an important technology. But can we do it securely? And what does "secure broadcast" mean? To us, secure broadcast revolves around two themes:

- receivers are certain that material they receive came from the appropriate sender; and
- senders have the option of limiting the recipients of particular messages.

In this chapter, we discuss these themes at length. And in this book, we address these concerns and many others, ranging from implementing secure broadcast on hardware with limited computational power to using broadcast mechanisms to execute denial-of-service attacks.

But first, let us begin by examining some basic terms. *Broadcast communication* is an essential mechanism for scalable information distribution. *Point-to-point communication* has been the dominant form of computer network communication since the beginning of networking. Unfortunately, with the explosive growth of information technology and the proliferation of the Internet and its applications, point-to-point communication faces serious scalability challenges. Distributing content, such as a popular movie, to a large audience over individual point-to-point connections is not economical. In contrast, one server can effortlessly reach vast audiences by using broadcast communication such as IP multicast.

A large number of broadcast applications exist today and even more are emerging. We present some broadcast applications here, and we discuss them in greater detail later in this chapter.

- **Content distribution over the Internet**. Digital TV and digital radio distributed over the Internet are becoming increasingly popular. Most likely, we will receive the future Soccer World Cup final or the opening ceremony of the Olympic games over an Internet or satellite broadcast.

- **Software distribution**. Broadcast enables quick and wide-spread distribution of software updates. For example, consider anti-virus software. Viruses utilize the Internet to infect hundreds of thousands of PC's in a few hours. As a countermeasure, an anti-virus center could broadcast virus scanner updates, and the operating system manufacturer could broadcast security patches.

- **Sensor networks**. The miniaturization of networked sensors brings an opportunity to solve many hard problems. Consider the problems of real-time road monitoring, real-time building safety monitoring (e.g., seismic safety), or fine-grained climate control in buildings. Since typical sensor networks communicate over wireless networks, broadcast is a natural communication method.

- **Transportation control**. By nature, transportation information affects multiple parties and is rapidly changing. Consider air traffic control. The majority of the communication patterns are broadcasts: GPS signals, airport landing system beacons, radio communications. Another example is road traffic control. Future cars will be equipped with wireless communication for driving support. Future highways may be equipped with beacons that broadcast location and current road information to cars.

- **Logistics and fleet monitoring**. Many companies need to track their fleet and can benefit from knowing accurate position information about their vehicles. Consider a taxi cab company or an express delivery service. With accurate position information, the coordination center can take informed decisions when it plans cab availability, or delivery times. An effective method for communication in such a setting is through broadcast. Location tracking is also useful for public services, such as buses and trains. Future buses may broadcast their current location and schedule to help passengers to predict the arrival time.

- **Personal wireless communications**. With the proliferation of cell phones we expect to see a rapid growth of broadcast-based wireless services and applications. For example, traffic monitors built into highways broadcast the current traffic pattern, public services such as hospitals and pharmacies may broadcast health-related information to people wearing health-support equipment. The potential applications are innumerable.

- **Home automation**. If the vision of ubiquitous or pervasive computing becomes a reality, almost every object within a household will be able to compute and communicate with other objects. Wireless communication (i.e., broadcast) may be an efficient and simple method for managing and querying these objects.

- **Financial markets**. Disseminating real-time financial market information efficiently to a large audience is an important challenge. We want to broadcast the information to all receivers simultaneously to attempt to achieve the goal of having receivers get the information at the same time, while protecting the integrity of the information.

- **Military applications**. The military has a wide range of applications that rely on broadcast to achieve robustness and survivability. Furthermore, many military applications rely on wireless communication.

- **Multi-player games**. Since the early days of networking, multi-player games are popular entertainment applications. Today, some games involve thousands of people simultaneously interacting in the virtual game world. Broadcast communication enables scalable and efficient games.

1.1 Challenges of Broadcast Communication

Point-to-point communication protocols are designed for one sender and one receiver. Unfortunately, the majority of point-to-point protocols do not generalize to broadcasting data to multiple receivers. Namely, broadcast communication encounters the following major challenges:

- **Reliability**. In point-to-point communication, a receiver achieves reliability by detecting missing or corrupted data, and requesting the sender to retransmit. In large-scale broadcasts, such an approach would not scale because a single lost packet can cause a flood of retransmission requests at the sender (this problem is sometimes called NACK implosion).

- **Receiver heterogeneity**. Some receivers may have high-bandwidth network connections and powerful workstations while others may have low-bandwidth connections with minimal computation resources.

- **Congestion control**. If a link in the Internet is congested, all well-behaved flows should back off until the link is not congested any more. Congestion control for IP multicast is particularly challenging.

- **Security**. Traditional security protocols for point-to-point communication suffer from the following problems in a broadcast setting: they may not scale to large audiences, they may not be secure, they may not be efficient and have a high computation or communication overhead, or they may not be robust to packet loss. We discuss these problems in more detail in Section 1.2.

Since reliability for individual packets is difficult to achieve in large-scale broadcasts, many broadcast applications distribute individual packets *unreliably*. Many broadcast applications do not expect reliability guarantees from the communication protocol, and the sender is not responsible for retransmitting lost packets. Thus, broadcast applications deal with packet loss on the application layer, e.g., multimedia applications experience quality degradation, or file transfer applications use forward error correction (FEC) [BLMR98].

In this book, we consider broadcast communication with the following properties: the sender *unreliably* distributes *real-time data*, the receiver wants to *immediately use data* as it arrives, and the broadcast must be secure. This is an especially challenging scenario, although common for many broadcast applications. Other broadcast settings may be simpler, and thus our protocols can easily provide security in those settings as well. To summarize, we consider broadcasts with the following features and requirements:

- Large numbers of receivers

- Receiver heterogeneity in computation resources (processor, memory, disk), and network resources (bandwidth, delay, reliability)

- The sender cannot retransmit lost packets

- Real-time data: the sender does not know the data in advance

- Streamed data: the receiver uses all data it receives
- Fast sending rate
- Security, in particular data authenticity and confidentiality

Applications with these requirements include real-time video streams, and data distribution that uses FEC [BLMR98, DF02, RMTR02]. We view FEC distribution as real-time data, as these systems often encode a fixed file into a long data stream where all packets are different (since these streams have a very long period, the server cannot pre-compute the stream in advance).

1.2 Why is Security for Broadcasts Hard?

In this section we analyze why efficient security protocols are challenging to design for broadcast environments. In contrast, a variety of protocols exist that provide efficient and secure protocols for authentication, signature, or confidentiality in point-to-point communication: SSL [FKK96a] and the standardized successor protocol TLS [DA99], and IPsec [KA98b, KA98a].

We now consider the security requirements of real-world broadcast applications and we investigate why these security requirements are much harder to achieve for broadcast communication with untrusted receivers, than in point-to-point communication.

1.2.1 Broadcast Authentication

The power of broadcast is that one packet can reach millions of receivers. This great property is unfortunately also a great danger: an attacker that sends one malicious packet can reach millions and just a single malicious network packet may be enough to cause a computer to lock up or reboot. An example of such an attack is the Windows Teardrop attack [CER97], which causes computers with the Windows NT operating system to halt.

This illustrates the importance of authentication. Unfortunately, efficient broadcast authentication is a challenging problem.

In the two-party (point-to-point) communication case, we can achieve data authentication through a purely symmetric mechanism[1]: the sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the receiver is assured that the sender generated that message.

[1]We explain *symmetric cryptography* and *message authentication codes* (MAC) in Section 2.3.

Symmetric MAC authentication is not secure in a broadcast setting, where receivers are mutually untrusted. The symmetric MAC is not secure: every receiver knows the MAC key, and could thus impersonate the sender and forge messages to other receivers. Intuitively, we need an asymmetric mechanism to achieve authenticated broadcast, such that every receiver can *verify* the authenticity of messages it receives, without being able to *generate* authentic messages.[2]

The property we seek is asymmetric, so it is natural to consider asymmetric cryptography, for instance, a digital signature. Digital signatures have the required asymmetric property: the sender generates the signature with its private key, and all receivers can verify the signature with the sender's public key. A digital signature provides non-repudiation, which is a much stronger property than authentication. Unfortunately, digital signatures have a high cost: they have a high computation overhead for both the sender and the receiver, as well as a high communication overhead. Since we assume broadcast settings where the sender does not retransmit lost packets, and the receiver still wants to immediately authenticate each packet it receives, we would need to attach a digital signature to each message. Because of the high overhead of asymmetric cryptography (as we show in Section 2.3), this approach would restrict us to low-rate streams and senders and receivers with powerful workstations. To deal with the high overhead of asymmetric cryptography, we can try to amortize one digital signature over multiple messages. We discuss this in Section 1.2.2. However, such an approach is still expensive in contrast to symmetric cryptography, since symmetric cryptography is in general 3 to 5 orders of magnitude more efficient than asymmetric cryptography.

To achieve higher efficiency with asymmetric cryptography, a sender could use short keys during a short time period (too short for an attacker to break the key with high probability), and to keep the sender and receivers time synchronized. Unfortunately, such an approach has many drawbacks. Consider a system that uses the RSA digital signature algorithm with 512-bit long keys, where the sender signs every packet with the current private key [RSA78]. Assume that every public/private key pair is valid for 5 minutes. Using a 512-bit long RSA key is probably about the smallest permissible key size today (even for short-time usage applications), as those keys can be factored today using only 2% of the computation required to break a 56 bit DES key [CWI99].

[2]In fact, Boneh, Durfee, and Franklin show that a general broadcast authentication protocol can be converted into a signature protocol [BDF01].

With today's resources and factoring technology, such a key can be factored in a few hours. The advantage is that the receiver can immediately verify the signature of the packet right after reception (i.e., no authentication delay). This approach also provides robustness to packet loss, as long as the receiver has a reliable mechanism to get the current public verification key. Unfortunately, this approach also has many drawbacks:

- **High computation overhead**. Despite the short key, a 800 MHz Pentium III workstation today can only sign about 640 messages per second, and verify about 6400 messages per second. For most applications, this would tie up the majority of computation resources. The computation overhead is overwhelming on smaller architectures. For example, a Palm Pilot or RIM pager can only generate between 0.17 and 0.4 signatures per second, and verify between 1.6 and 10 signatures per second [BCH$^+$00].

  Despite recent progress in digital signatures, RSA still provides one of the fastest signature verification for digital signatures based on number-theoretic assumptions.

- **High communication overhead**. With a 512 bit RSA key, each signature is 64 bytes long, which is too long for many applications. Recent signature algorithms have shorter signatures, but most of them have a higher computation overhead than RSA [CGP01, LV00]. Furthermore, the sender needs to update the public key every 10 minutes, which has an additional overhead.

- **Long-lived devices**. Some of the devices will be deployed for several years. Unfortunately, 512 bit long RSA keys will not be secure as factoring algorithms advance and processors continue to double in speed every 18 months.

- **Not perfectly robust to packet loss**. If the packets that carry the new public key are lost, the receivers cannot authenticate subsequent packets.

- **Time synchronization requirement**. The receivers need to be loosely time synchronized with the sender; however, the time synchronization error can be on the order of minutes.

For efficient broadcast authentication, we design two protocols that rely on purely symmetric cryptography: the TESLA protocol (Chapter 3) and the BiBa broadcast authentication protocol (Chapter 4). Both protocols require time synchronization between the sender and the receiver. Furthermore, both protocols use novel techniques to provide the asymmetric property required for broadcast authentication. TESLA achieves the asymmetry through time delayed key disclosure, and BiBa is a new combinatorial signature.

To summarize, a viable broadcast authentication protocol must have the following properties:

- Low computation overhead for generation and verification of authentication information.
- Low communication overhead.
- Limited buffering required for the sender and the receiver (timely authentication for each individual packet).
- Strong robustness to packet loss.
- Scales to a large number of receivers.

1.2.2 Broadcast Signature

Some applications require that each message of the broadcast stream is digitally signed, to provide non-repudiation of origin in a court of law, for example stock market data, on-line auctions. Digital signatures require an asymmetric property, such that the signer can generate a signature, and the receivers can only verify (i.e., not generate) the signature. Compared to symmetric cryptography, asymmetric cryptographic primitives are orders of magnitude slower as we show in Section 2.3. Hence, if the receiver needs to compute an asymmetric cryptographic primitive for each packet it receives, the receiver needs substantial computation power. A more efficient approach is to amortize one asymmetric operation over multiple packets. We present the EMSS and MESS protocols which amortize one digital signature over thousands of packets.

The requirements are similar to the requirements of broadcast authentication. The properties of an ideal signature protocol for broadcast streams are:

- Low computation overhead for generation and verification of authentication information
- Low communication overhead
- No buffering required for the sender and the receiver
- Instant signature verification for each individual message
- Strong robustness to packet loss
- Scales to large number of receivers

1.2.3 Broadcast Data Integrity

Data integrity ensures that the data was not modified or deleted in any unauthorized way. In this work, we achieve integrity through authentication. Authentication ensures that the data originates from the claimed source, and that the data was not modified in transit. To detect unauthorized data deletion, we can use a variety of techniques, for instance adding a counter to data will allow the recipient to detect missing data. However, in most broadcast settings, the sender does not retransmit lost packets (or missing data) as we discuss in Section 1.1, so we do not need a counter. Hence, we achieve data integrity through data authentication.

1.2.4 Confidential Broadcasts and Restricting Access to Legitimate Receivers

Most contemporary broadcast environments allow any receiver to receive broadcast information. For instance, wireless environments allow anybody within the sending range to receive the broadcast. In IP multicast, for example, any receiver can subscribe to a multicast group. However, in many applications the sender wants to *restrict access* and to control which receivers can receive broadcast information, for example in subscription-based information distribution such as stock quote feeds, weather information, news, or sports broadcasts. The general approach for achieving this requirement is to encrypt broadcast information with a secret key $\mathcal{X}$. Only the sender and legitimate receivers know $\mathcal{X}$. Any illegitimate receiver that does not know $\mathcal{X}$ will not be able to decrypt the broadcast information. This approach also provides *confidentiality* of the broadcast information. So we can solve the access control problem for broadcast information if we can solve the key distribution problem.

Unfortunately, distributing a secret key efficiently to a large number of receivers is a challenge. The main problem is to update the key in dynamically changing groups, since a receiver should only be able to decrypt the broadcast stream while it is a member of the group. More concretely, the group key that the receiver gets when it joins the group should not allow it to decrypt content that was sent before it joined the group (we call this the *backward secrecy* property), and similarly, the member should not be able to decrypt any broadcast content after it leaves the group (we call this the *forward secrecy* property). *Group key secrecy* is another important property, which means that no outsider can find any group key. We discuss these properties in more detail in Section 6.1.1.

To achieve forward and backward secrecy, the sender updates the key after each member join or member leave event and sends the new group key to all legitimate receivers. Updating the group key in a *secure*, *scalable*, and *reliable* manner is challenging. In particular, achieving reliability is crucial in most current key distribution schemes, since a missing group key prevents a receiver from decrypting subsequent messages. Chapter 6 discusses these issues in more detail. To summarize, a viable key distribution protocol provides the following properties:

- Secure key update
  - Group key secrecy
  - Backward secrecy
  - Forward secrecy
- Scalability for large dynamic groups
- Reliability of key update messages
- Low computation overhead
- Low communication overhead (small key update messages)

1.3 Security Requirements for Broadcast Applications

We now discuss current and emerging broadcast applications and analyze their security requirements.

An important broadcast application is **software distribution**. Distribution of software over the Internet is increasing rapidly. For popular and large software packages, broadcast distribution is probably the most effective and economical distribution mechanism, in particular for security critical updates. For example, consider the dissemination of anti-virus software updates and operating system patches. Modern computer viruses and worms use the Internet to propagate, and can spread quickly and infect hundreds of thousands of connected computers within hours. For instance, the recent Code Red worm infected over a quarter million Microsoft Windows NT servers within hours [CER01a, CER01b]. To counteract these worms, we could distribute anti-virus software updates or operating-system patches through broadcasts, with the hope that we can protect the computers before they are attacked. The important security requirement for software distribution is that all receivers must authenticate the origin of the data, such that an adversary could not distribute rogue code. It may appear that a digital signature on the entire data suffices. In practice, however, the sender usually uses a technique such as forward error correction (FEC) to achieve reliable content distribution [BLMR98, DF02, RMTR02]. These techniques encode the data into many small chunks, the receivers collect the chunks, and reconstruct the data. If an adversary injects a single bad chunk, the reconstruction fails, or produces garbled data and the signature verification would fail. Hence, injecting bogus packets poses a serious risk for a denial-of-service attack. Due to the exceedingly large number of combinations, it is computationally infeasible for the receiver to try to guess good chunks and attempting to reconstruct the data until the digital signature matches. A better approach is to use authentication on each chunk, so the receiver can authenticate the origin of every individual chunk before it further processes it.

An additional requirement may be to restrict access, so only legitimate receivers receive the data, which is necessary in many software distribution settings. We discuss the access control issue further in Section 1.2.4.

Another important application that requires broadcast communication is the distribution of **audio and video data**. Due to server limitations, current network bandwidth limits, and the relatively high bandwidth requirements of high-fidelity video streams, popular content requires broadcast for scalability. For instance popular sports events (e.g., the soccer world cup final), international events (e.g., the opening ceremony of the Olympic games), new video clips by music celebrities, or important political speeches, may attract up to hundreds of millions of viewers. Such events would require a tremendous server infrastructure to distribute the information by point-to-point communication. In most applications each receiver needs to authenticate the data origin of each message it receives to prevent an attacker from sending bogus content and potentially hijacking the entire stream. In settings where the information distributor restricts access to receivers who pay for the content, the sender may use access control techniques, which we discuss further in Section 1.2.4.

Audio and video broadcasts are also important in corporate environments, for instance for employee training and information distribution. Content authentication may not be necessary within a closed corporate network. Restricting access to legitimate receivers, however, may be necessary for sensitive and competition-critical information.

Another important broadcast application is **real-time stock market quote feeds**. Distributing the current market information such that millions of receivers get the data simultaneously is a challenge. Security is also an important requirement in this setting: data authenticity is paramount for the receivers, and the information distributors want confidentiality, such that only the paying receivers get the information.

Sensor networks are an emerging technology. Sensors may monitor a wide variety of events, such as sound, temperature, light, cars passing by, motion, water level, etc. The number of applications are endless. These sensors usually communicate over a wireless channel, hence broadcast is a natural communication method. Because wireless communication is easy to eavesdrop or to spoof a message, many sensor networks require confidentiality and authentication. We discuss sensor networks and their applications in detail in Chapter 7.

The **air traffic control** system used today relies heavily on broadcast: GPS signals, airport landing system signals, and radio communications. Unfortunately, the infrastructure deployed today does not use cryptographic methods to protect against eavesdropping or fake signals. To guarantee maximum security for the flights, air traffic control should at the least use strong authentication on all broadcast signals, and if possible, restrict access to legitimate receivers.

1.4 Novel Contributions

This book introduces the following novel contributions:

- We propose to use time to achieve the asymmetry necessary in broadcast authentication (receivers can only verify, but not generate any valid authentication information). The TESLA broadcast authentication protocol achieves asymmetry through delayed key disclosure. Through the use of one-way key chains or Merkle hash trees, we achieve perfect robustness to packet loss. The TIK extension exploits precise time synchronization and enables receivers to instantly authenticate a packet. In case of loosely synchronized clocks, we introduce TESLA-IA (TESLA with instant authentication), where we shift from receiver-side buffering to sender-side buffering.

- We introduce use of multi-way collisions for a signature. This is a novel approach for constructing signatures based on one-way functions without trapdoors. The resulting signature provides low verification overhead. We design the BiBa broadcast authentication protocol based on the BiBa signature, which achieves a previously unachieved set of properties (instant authentication, scalability, robustness to packet loss, efficient authentication, reasonable communication overhead, loose time synchronization).

- The EMSS/MESS stream signature protocols use multiple hash links to achieve robustness to packet loss. The HTSS stream authentication protocol features a previously unachieved low communication overhead with instant authentication. This efficiency stems from observing the sending order of packets, and nodes of the Merkle hash tree; and by minimizing the redundancy of sending the Merkle hash tree nodes.

- The ELK large-group key distribution protocol introduces several novel ideas. The ELK protocol reduces the broadcast communication overhead for group key updates and introduces several mechanisms for achieving reliability. ELK introduces a protocol that does not require any broadcast message after a member joins the group at any time, or after a member leaves the group at a predicted time. A novel key update protocol enables us to reduce the key size, without introducing a vulnerability to dictionary attacks. The key update protocol also enables us to compress key update messages, by exploiting receiver computation. Finally, we reduce the size of redundant key update packets (for achieving robustness to packet loss) by selecting only the key update components which are useful for the majority of group members.

- As a case study, we consider sensor networks with highly resource-starved devices. Despite the hardware constraints, we design secure protocols for point-to-point communication, and adapt the TESLA broadcast authentication protocol to this environment.

1.5 Scope of this Book

This book presents a number of new protocols to secure broadcast communication. The emphasis of the text is on the new mechanisms that these protocols introduce. We describe our mechanisms at a fairly high level, so that the description is general enough to apply to a wide variety of settings. We omit detailed security proofs of our mechanisms, but we give general security arguments and refer to publications containing more detailed security arguments.

1.6 Book Overview

This book presents new methods and protocols for secure broadcast communication. Rather than present protocols for specific settings, we present a flexible framework and a set of building blocks appropriate to a wide variety of settings. To ensure generality of our methods, we intentionally leave out specific implementation details. However, we do consider specific applications for each protocol, we do walk through a specific solution, and we do discuss performance and tradeoffs. The tools in this book support secure protocols for any broadcast network, including IP multicast, wireless networks, and satellite distribution networks. Here is an outline of this book.

- Chapter 1 is this introduction. We present applications for broadcast networks and discuss their security requirements. We identify broadcast authentication, broadcast signature, and key distribution as the essential and fundamental security requirements for broadcast networks.
- Chapter 2 introduces the basic cryptographic primitives that we use.
- Chapter 3 presents the TESLA protocol, a highly efficient broadcast authentication protocol. TESLA uses time (delayed key disclosure) to achieve the asymmetry property required for secure broadcast authentication. The main features of TESLA are: low sender and receiver computation overhead (around one MAC function computation per packet), low communication overhead, and perfect robustness to packet loss. TESLA requires loosely synchronized clocks between the sender and the receiver, and the authentication is slightly delayed. However, we describe a variant of TESLA with instant authentication, which requires sender buffering instead of receiver buffering, but the receiver can authenticate the message as soon as the packet arrives.
- Chapter 4 presents the BiBa signature algorithm, a new approach to design signature algorithms based on one-way functions without a trapdoor (these signatures are sometimes called one-time signatures). BiBa is an acronym for bins and balls signature. We use bins and balls as an analogy to describe BiBa: to sign a message, the signer uses the message to seed a random process which throws a set of balls into bins. When enough balls fall into the same bin, the combination of those balls constitute a signature. To achieve an asymmetry between the signer and the forger, the BiBa signature exploits the property that the signer who has a large number of balls finds a signature with high probability, but a forger who only has a small number of balls has a negligible probability to find a signature. To the best of our knowledge, the BiBa signature is one of the fastest signatures today for verification. Signature generation is more expensive, but only requires two sequential hash function computations if the signer has many parallel processors. BiBa is about twice as fast than most previous one-time signa-

ture algorithms, and the signature size is less than half as large than most previous signatures based on one-way functions without trapdoors.

Using the BiBa signature as a basis, we design the BiBa broadcast authentication protocol. BiBa requires loose time synchronization between the sender and receivers, but in contrast to TESLA, the authentication is instant and neither the sender nor the receiver need to buffer messages. BiBa is also perfectly robust to packet loss, and scales well. The BiBa broadcast authentication protocol thus achieves a unique set of properties.

TESLA and BiBa feature new approaches to achieve the asymmetric property required by the broadcast authentication: TESLA uses time delay, and BiBa is a new combinatorial signature.

- Chapter 5 introduces a set of broadcast signature protocols: EMSS, MESS, and HTSS. EMSS and MESS use sequences of hashes to amortize one expensive digital signature operation over many messages. In a nutshell, the sender computes the hash of a packet and adds it to later packets. Periodically, the sender signs data packets with a digital signature. Due to the sequence of hashes, the signature extends transitively over the previous packets, i.e., the receiver can follow the sequence of hashes to verify previous packets. Because the hash of a packet appears redundantly in later packets, we achieve tolerance to packet loss. Given a loss probability, we analyze in how much redundancy we need to achieve a high probability that we can verify the signature of a packet. For the special case where the sender knows the entire stream content in advance, we design the broadcast signature protocol HTSS. HTSS constructs a hash tree over all the messages and the sender only needs to sign the root of the tree. Naively, such an approach would seem to require high communication overhead, but we present an efficient method of encoding to make this approach viable.

  Table 1.1 shows an overview of the features of our broadcast authentication and signature protocols presented in Chapters 3 through 5. To select an appropriate protocol, we can choose from three broadcast signature protocols if we need non-repudiation: EMSS, MESS, HTSS. However, many broadcast applications require authentication, so TESLA is ideally suited if loose time synchronization is possible and a short authentication delay is tolerable. If time synchronization is less accurate, or if we want the lowest possible end-to-end authentication delay, the BiBa broadcast authentication protocol is appropriate. If the data may be cached for extended time peri-

	TESLA	TESLA-IA	TIK	BiBa	EMSS / MESS	HTSS
Authentication (A) or signature (S)	A	A	A	A	S	S
Time synchronization	Y	Y	Y	Y	N	N
Authentication delay	Y	N	N	N	Y	N
Sender buffering	N	Y	N	N	N	Y
Receiver buffering	Y	N	N	N	Y	N
Real-time streams	Y	Y	Y	Y	Y	N
Robustness to packet loss	Y	N	Y	Y	N	N
Communication overhead (bytes)	24	32	24	128	40	32
Generation overhead	1	1	1	2048	1	2
Verification overhead	2	2	30	100	1	10

Table 1.1. This table compares the authentication and signature schemes we propose. TESLA is the basic protocol we describe in Chapter 3. TESLA-IA is the instant authentication extension we present in Section 3.5.1. BiBa is the stream authentication protocol we present in Chapter 4. Chapter 5 presents the EMSS, MESS, and HTSS stream signature schemes. The communication overhead row lists the approximate per-packet size of the authentication or signature information. The unit for the generation and verification overheads is the approximate number of hash function computations.

ods and time synchronization is not possible, we suggest use of one of the signature protocols to provide authentication.

- Chapter 6 discusses the problem of how to distribute a key securely and reliably to large numbers of receivers, in a scalable manner. We identify reliability as a crucial factor to achieve scalability, and design multiple protocols that greatly enhance reliability. For instance, we design a new protocol for adding a new receiver to the group that does not require any broadcast message. We also design a new protocol that updates the group key after a member leaves the group at a predicted time, without sending a broadcast message. These two protocols greatly enhance scalability, because fewer messages need to be sent reliably. We furthermore present a new key update protocol that features compressed key update messages. Our protocol enables such short key update messages that the sender can send them along with regular data messages. We present the ELK protocol, which uses all these protocols to achieve a highly efficient and scalable key distribution protocol.

- Chapter 7 examines a case study in an extremely resource-constrained sensor network. We analyze the security requirements of sensor networks. We present the Sensor Network Encryption Protocol (SNEP). SNEP secures point-to-point communication links and provides confidentiality, authentication, and freshness. Broadcast is a natural method of communication in wireless networks, and authentication of broadcast messages is important for many applications. By designing the μTESLA protocol, we demonstrate that the TESLA protocol can scale down to minimal environments and that broadcast authentication is practical even in sensor network environments.

- Chapter 8 reviews related work.

- Chapter 9 concludes this book. We end with a discussion of open research problems in the area of secure broadcast.