Roll. No. A016	Name: Varun Khadayate
Class B.Tech CsBs	Batch: 1
Date of Submission: 25-10-2022	Subject: Cryptology

Zero Knowledge Protocol

Zero knowledge proof or protocol is method in which a party A can prove that given statement X is certainly true to party B without revealing any additional information.

Let's say Alice and Bob want to communicate over shared network. Alice initiates the communication and sends secret to Bob. Bob verifies the secret so he can be certain that he is communicating with Alice. Once he verifies the secret, he sends conformation. In the above scenario, Bob must know Alice's secret so he can verify Alice's identity but now Bob can impersonate Alice.

Zero knowledge protocol allow Alice to prove Bob that she knows the secret without revealing the secret. In this protocol, verification is performed for many executions and each time Alice need S to pass the verification.

Zero knowledge protocol is 3 pass identification protocol. First message is Commitment or witness sent from Alice to Bob, a second message is challenge sent from Bob to Alice and a final message is response sent from Alice to Bob.

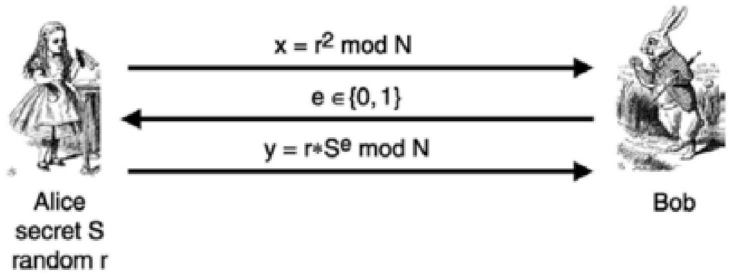
Zero knowledge protocol must have three properties.

1. Completeness: If the statement is true, the honest verifier will be convinced by honest prover.
2. Soundness: If the statement is false, Trudy cannot convince the verifier that it is true, except with some small probability.
3. Zero-knowledge: If the statement is true no cheating verifier learns anything other than this fact.

Randomness is also an important property of Zero knowledge protocol. Randomness in the commitment and challenge message are used to hide the secret information.

Fiat-Shamir Protocol

- Goal of protocol is to prove Alice knows secret s in n executions.
- This is probabilistic protocol with probability of for adversary to fool the verifier.
- Usually, the number of executions are around 20 to 40



One time Set up:

1. In this protocol, Trusted center selects RSA like modulus N = p* q , where p and q are secret prime number and N is public.
2. Alice selects secret S such that S is coprime to N and I<=S<=N-I, computes V = S^2 mod N. S is private, and V is public.

Protocol:

1. Alice selects random number r and sends x = r^2mod N to Bob.
2. Bob sends either O or I to Alice.
3. Alice sends y = r * s^e mod N to Bob.

Verification:

1. Bob verifies it with Y2 = x * v^e mod N.

Pros:

- Secured — Not requiring the revelation of one's secret.
- Simple — Does not involve complex encryption methods.

Cons:

- Limited — Secret must be numerical, otherwise a translation is needed.
- Lengthy — There are 2k computations, each computation requires a certain amount of running time.
- Imperfect — The Malice can still intercept the transmission (i.e., messages to the Verifier or the prover might be modified or destroyed).

Applications of Zero Knowledge Protocol:

1. *Authentication Systems*
   The need for authentication systems where one party uses a secret piece of information (like a password) to prove its identity to a second party without disclosing the secret to the second party has sparked research in zero-knowledge proofs. But in many techniques for zero-knowledge proofs of knowledge, a password is usually too short or not sufficiently random. A unique type of zero-knowledge proof of knowledge that addresses the restricted length of passwords is the zero-knowledge password proof. In April 2015, the Sigma protocol (one-out-of-many proofs) was introduced. In August 2021, Cloudflare, an American web infrastructure and security company decided to use the one-out-of-many proofs mechanism for private web verification using vendor hardware.
2. *Ethical Behaviour*
   Zero-knowledge proofs are a tool used in cryptographic systems to enforce moral conduct while preserving anonymity. A user is essentially required to demonstrate using a zero-knowledge proof that its activity is appropriate in accordance with the protocol. We know that the user must truly act honestly to be able to offer a legitimate proof because of soundness. We know that the user does not violate the privacy of its secrets in the process of submitting the proof because we have no prior information.
3. *Nuclear Disarmament*
   The Princeton Plasma Physics Laboratory and Princeton University demonstrated a method in 2016 that might be used in upcoming negotiations on nuclear disarmament. It would enable inspectors to verify whether a piece of equipment contains a nuclear weapon without recording, sharing, or disclosing any potentially sensitive internal operations.
4. *Blockchains*
   The Zerocoin and Zerocash protocols, which culminated in the creation of Zcoin (later rebranded as Firo in 2020) and Zcash cryptocurrencies in 2016, used zero-knowledge proofs. To maintain anonymity, Zerocoin offers a built-in mixing strategy that does not trust any peers or centralised mixing providers. Users have the option to deal in a base currency and cycle it in and out of Zerocoins. While Zerocoin cannot conceal the transaction amount, the Zerocash protocol uses non-interactive zero-knowledge proof. Zerocash is less vulnerable to privacy timing attacks than Zerocoin due to the network's severe limits on transaction data. However, since fraudulent coins cannot be traced, this additional layer of anonymity could lead to an undetected hyperinflation of the Zerocash supply. Bulletproofs were first presented in 2018. Compared to non-interactive zero-knowledge proofs, when trusted setup is not required, bulletproofs are an enhancement. Later, it was included to the Monero cryptocurrency and the Mimblewimble protocol, upon which the cryptocurrencies Grin and Beam are based. The Sigma protocol, an upgrade to the Zerocoin protocol without trusted setup, was deployed by Firo in 2019. The Lelantus protocol, which Firo introduced in the same year, is an upgrade over the Sigma protocol in that it conceals a transaction's source and value.

Key Management

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key.

For two parties A and B, key distribution can be achieved in several ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third-party C, C can deliver a key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link. However, for end-to-end encryption over a network, manual delivery is awkward. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time. Thus, each device needs a number of keys supplied dynamically. The problem is especially difficult in a wide-area distributed system.

For end-to-end encryption, some variation on option 4 has been widely adopted. In this scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed. Each user must share a unique key with the key distribution center for purposes of key distribution.

The use of a key distribution center is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used (Figure 14.2). Communication between end systems is encrypted using a temporary key, often referred to as a session key. Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded. Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a master key that is shared by the key distribution center and an end system or user.

For each end system or user, there is a unique master key that it shares with the key distribution centers. Of course, these master keys must be distributed in some fashion. However, the scale of the problem is vastly reduced. If there are N entities that wish to communicate in pairs, then, as was mentioned, as many as [N (N - 1)]/2 session keys are needed at any one time. However, only N master keys are required, one for each entity. Thus, master keys can be distributed in some non-cryptographic way, such as physical delivery.

It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so. As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building. For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC. In this case, any one of the three KDCs involved can actually select the key.

The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork. A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities. Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key. On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

For connection-oriented protocols, one obvious choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session. If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles.

For a connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination. Thus, it is not obvious how often one needs to change the session key. The most secure approach is to use a new session key for each exchange. However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction. A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized. Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.

A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution. Thus, there may need to be as many as $[n\ (n-1)]/2$ master keys for a configuration with n end systems.

A session key may be established with the following sequence of steps
(Figure 14.5).

1. A issues a request to B for a session key and includes a nonce, N_1.
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value $f(N_1)$, and another nonce, N_2.
3. Using the new session key, A returns $f(N_2)$ to B.

Thus, although each node must maintain at most $(n - 1)$ master keys, as many
session keys as required may be generated and used. Because the messages transferred
using the master key are short, cryptanalysis is difficult. As before, session
keys are used for only a limited time to protect them.

The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed. It also may be desirable to impose some control on the way in which automatically distributed keys are used. For example, in addition to separating master keys from session keys, we may wish to define different types of session keys based on use, such as

- Data-encrypting key, for general communication across a network
- PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications
- File-encrypting key, for encrypting files stored in publicly accessible locations

Thus, it may be desirable to institute controls in systems that limit the ways in which keys are used, based on characteristics associated with those keys. One simple plan is to associate a tag with each key ([JONE82]; see also [DAVI89]). The proposed technique is for use with DES and makes use of the extra 8 bits in each 64-bit DES key. That is, the eight non-key bits ordinarily reserved for parity checking form the key tag. The bits have the following interpretation:

- One bit indicates whether the key is a session key or a master key.
- One bit indicates whether the key can be used for encryption.
- One bit indicates whether the key can be used for decryption.
- The remaining bits are spares for future use.

Because the tag is embedded in the key, it is encrypted along with the key when that key is distributed, thus providing protection. The drawbacks of this scheme are

1. The tag length is limited to 8 bits, limiting its flexibility and functionality.
2. Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled.

An extremely simple scheme was put forward by Merkle.

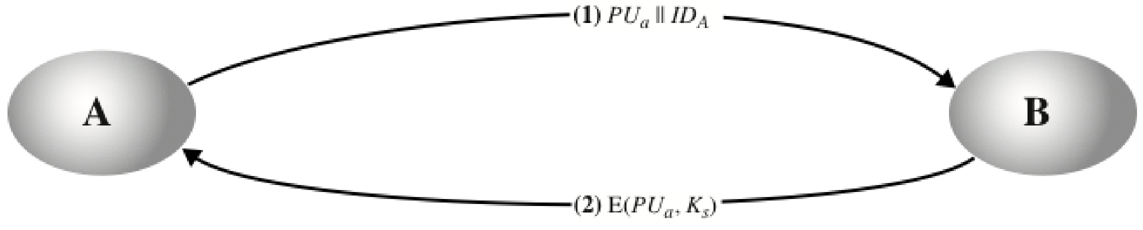


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.

The protocol depicted in Figure 14.7 is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message (see Figure 1.3c). Such an attack is known as a man-in-the-middle attack [RIVE84]. We saw this type of attack in Chapter 10 (Figure 10.2). In the present case, if an adversary, D, has control of the intervening communication channel, then D can compromise the communication in the following fashion without being detected (Figure 14.8).

Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

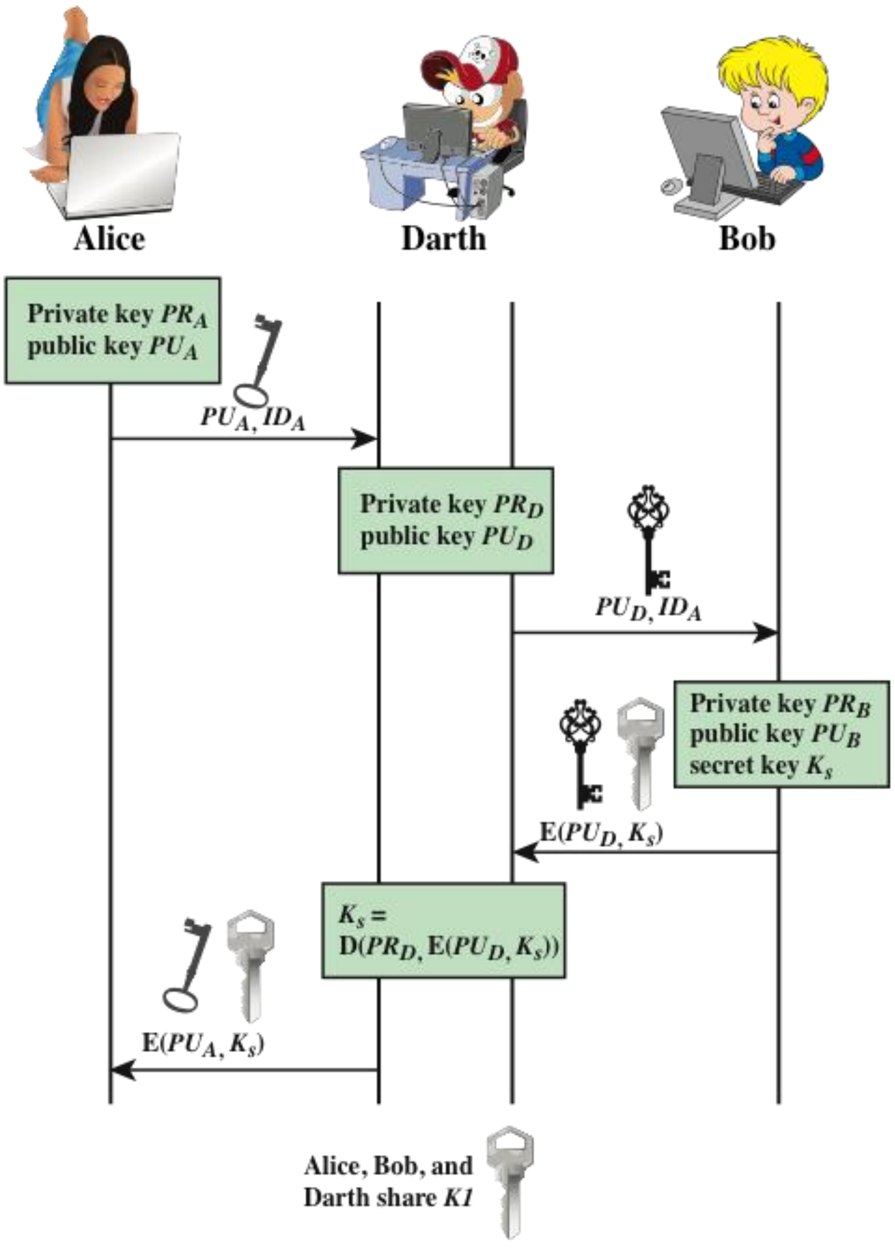


Figure 14.8 Another Man-in-the-Middle Attack

Figure 14.9, based on an approach suggested in [NEED78], provides protection against both active and passive attacks.

The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

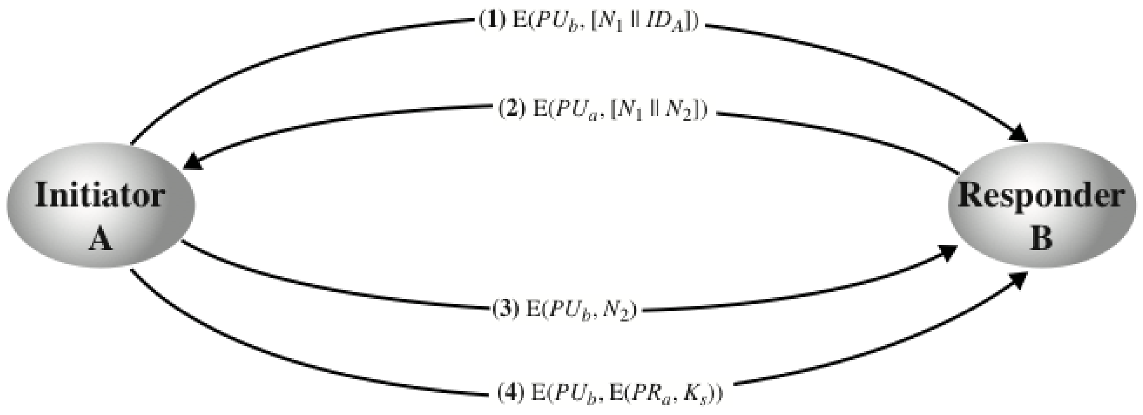

(1) $E(PU_b, [N_1 \| ID_A])$
(2) $E(PU_a, [N_1 \| N_2])$
Initiator A
Responder B
(3) $E(PU_b, N_2)$
(4) $E(PU_b, E(PR_a, K_s))$

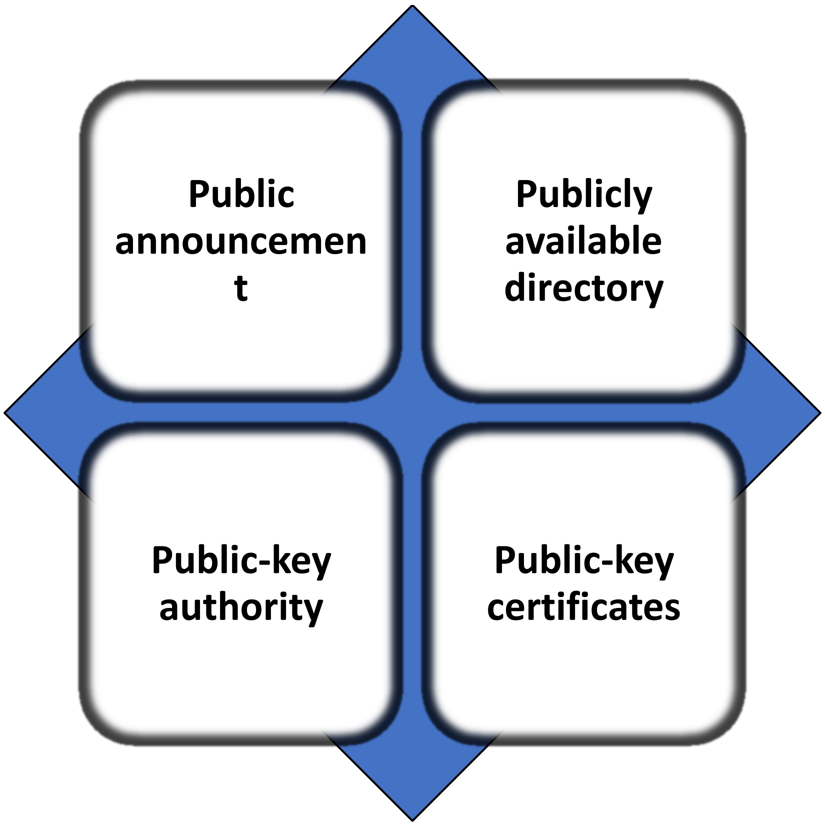
Figure 14.9 Public-Key Distribution of Secret Keys

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes [LE93]. This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key. A public-key scheme is used to distribute the master keys. The following rationale is provided for using this three-level approach:

- Performance: There are many applications, especially transaction-oriented applications, in which the session keys change frequently. Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.
- Backward compatibility: The hybrid scheme is easily overlaid on an existing KDC scheme with minimal disruption or software changes.

The addition of a public-key layer provides a secure, efficient means of distributing master keys. This is an advantage in a configuration in which a single KDC serves a widely distributed set of users.

Several techniques have been proposed for the distribution of public keys, which can mostly be grouped into the categories shown.



On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large (Figure 14.10). For example, because of the growing popularity of PGP (pretty good privacy, discussed in Chapter 19), which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET newsgroups and Internet mailing lists.

Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication (see Figure 9.3).

A
B
PU_a
PU_a
PU_a
PU_a
PU_b
PU_b
PU_b
PU_b

Figure 14.10 Uncontrolled Public Key Distribution

More security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization (Figure 14.11). Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

This scheme is clearly more secure than individual public announcements but still has vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

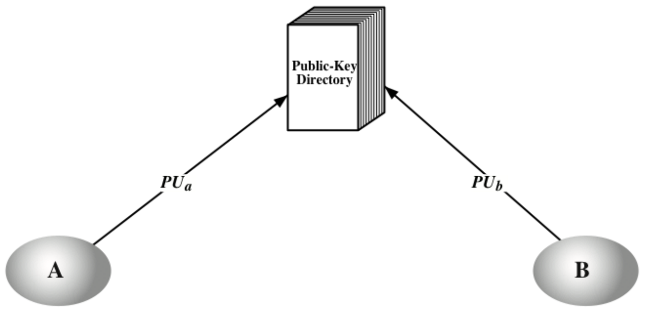


Figure 14.11 Public Key Publication

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure 14.12, which is based on a figure in [POPE79]. As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.

A total of seven messages are required. However, the initial five messages need be used only infrequently because both A and B can save the other’s public key for future use—a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

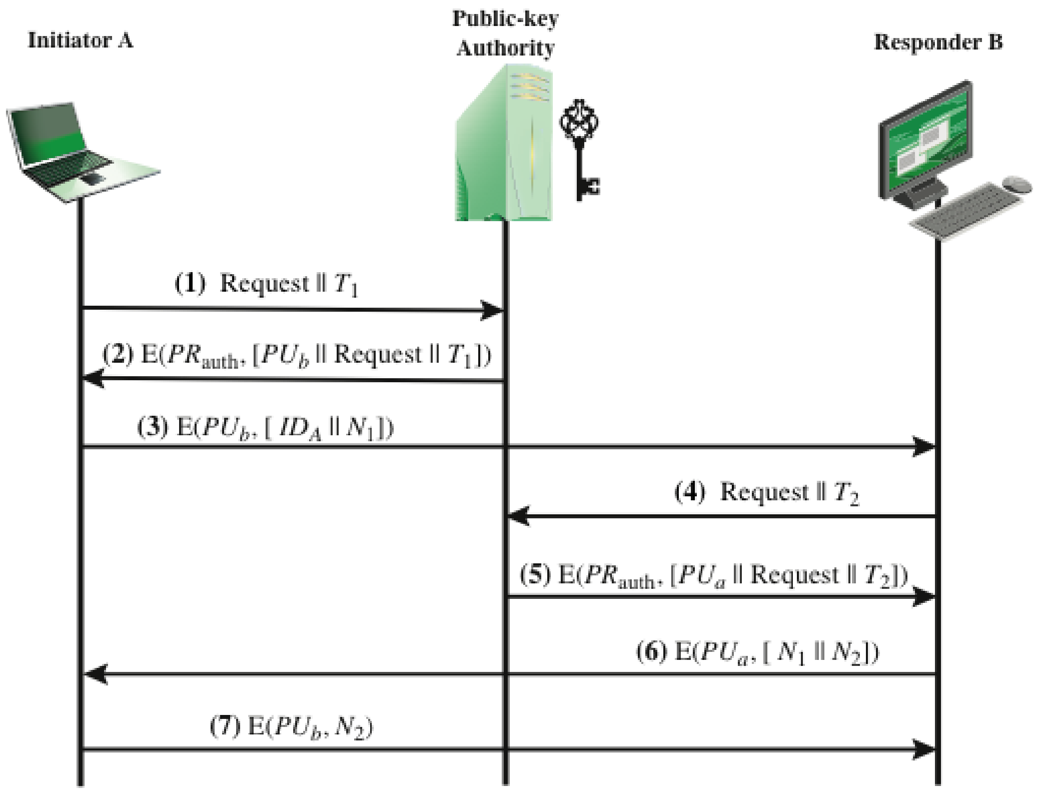

Initiator A
Public-key Authority
Responder B
(1) Request ‖ T_1
(2) E(PR_{auth}, [PU_b ‖ Request ‖ T_1])
(3) E(PU_b, [ID_A ‖ N_1])
(4) Request ‖ T_2
(5) E(PR_{auth}, [PU_a ‖ Request ‖ T_2])
(6) E(PU_a, [N_1 ‖ N_2])
(7) E(PU_b, N_2)

Figure 14.12 Public-Key Distribution Scenario

The scenario of Figure 14.12 is attractive, yet it has some drawbacks. The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

An alternative approach, first suggested by Kohnfelder [KOHN78], is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.

Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.

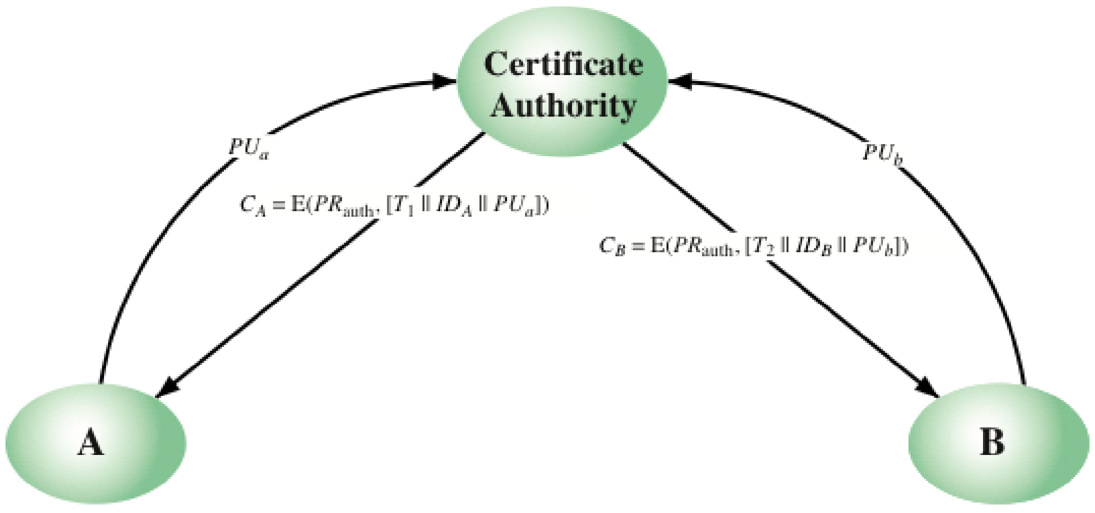
A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. We can place the following requirements on this scheme:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.

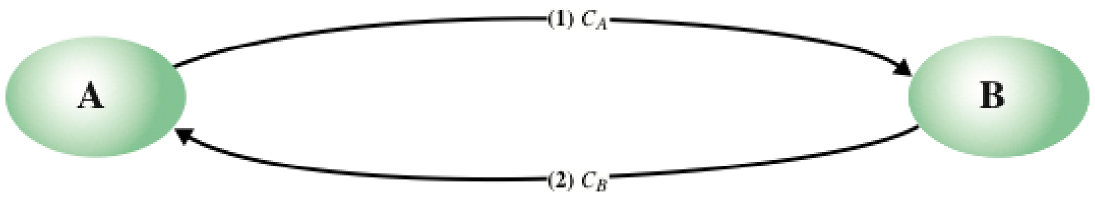
These requirements are satisfied by the original proposal in [KOHN78]. Denning [DENN83] added the following additional requirement:

4. Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in Figure 14.13. Each participant applies to the certificate authority, supplying a public key and requesting a certificate. Application must be in person or by some form of secure authenticated communication.


Certificate Authority
PU_a
PU_b
$C_A = \mathrm{E}(PR_{\mathrm{auth}}, [T_1 \| ID_A \| PU_a])$
$C_B = \mathrm{E}(PR_{\mathrm{auth}}, [T_2 \| ID_B \| PU_b])$
A
B

(a) Obtaining certificates from CA



(b) Exchanging certificates

Figure 14.13 Exchange of Public-Key Certificates

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific digital signature algorithm nor a specific hash function. Figure 14.14 illustrates the overall X.509 scheme for generation of a public-key certificate. The certificate for Bob's public key includes unique identifying information for Bob, Bob's public key, and identifying information about the CA, plus other information as explained subsequently. This information is then signed by computing a hash value of the information and generating a digital signature using the hash value and the CA's private key. X.509 indicates that the signature is formed by encrypting the hash value. This suggests the use of one of the RSA schemes discussed in Section 13.6. However, the current version of X.509 does not dictate a specific digital signature algorithm. If the NIST DSA (Section 13.4) or the ECDSA (Section 13.5) scheme is used, then the hash value is not encrypted but serves as input to a digital signature generation algorithm.

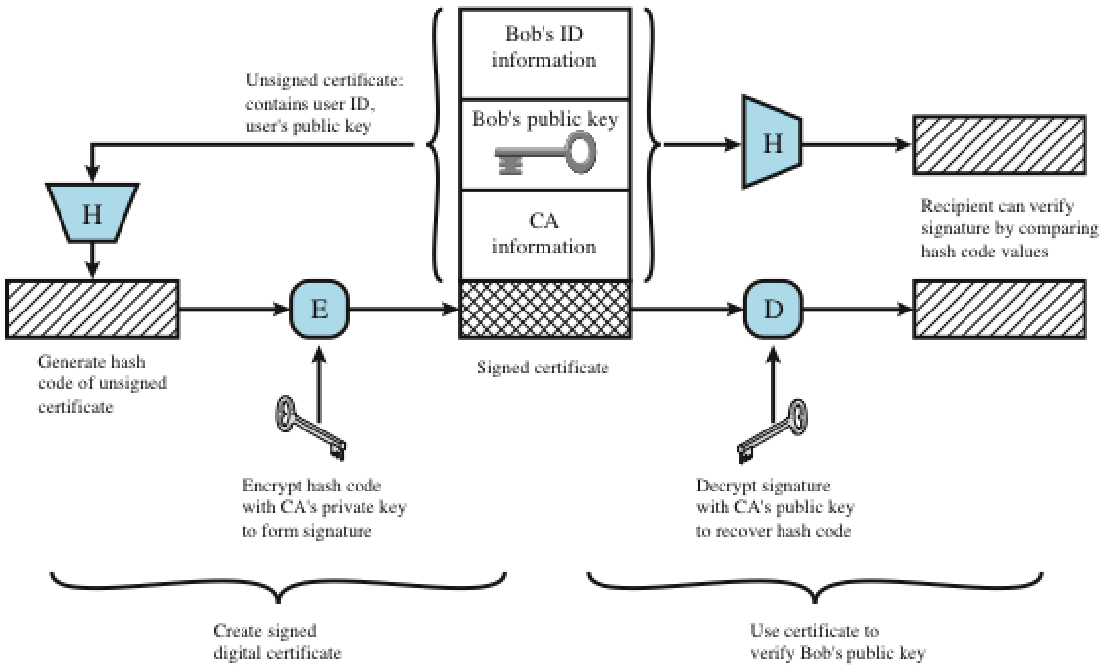

Bob's ID information
Bob's public key
CA information
Unsigned certificate: contains user ID, user's public key
H
H
E
D
Recipient can verify signature by comparing hash code values
Generate hash code of unsigned certificate
Signed certificate
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

Figure 14.14 Public-Key Certificate Use

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

- Version: Differentiates among successive versions of the certificate format; the default is version 1. If the issuer unique identifier or subject unique identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
- Serial number: An integer value unique within the issuing CA that is unambiguously associated with this certificate.
- Signature algorithm identifier: The algorithm used to sign the certificate Together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.
- Issuer name: X.500 name of the CA that created and signed this certificate.
- Period of validity: Consists of two dates: the first and last on which the certificate is valid.
- Subject name: The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- Subject’s public-key information: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- Issuer unique identifier: An optional-bit string field used to identify uniquely the issuing CA in

the event the X.500 name has been reused for different entities.

- Subject unique identifier: An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.
- Extensions: A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- Signature: Covers all the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier. The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time. These fields are rarely used.

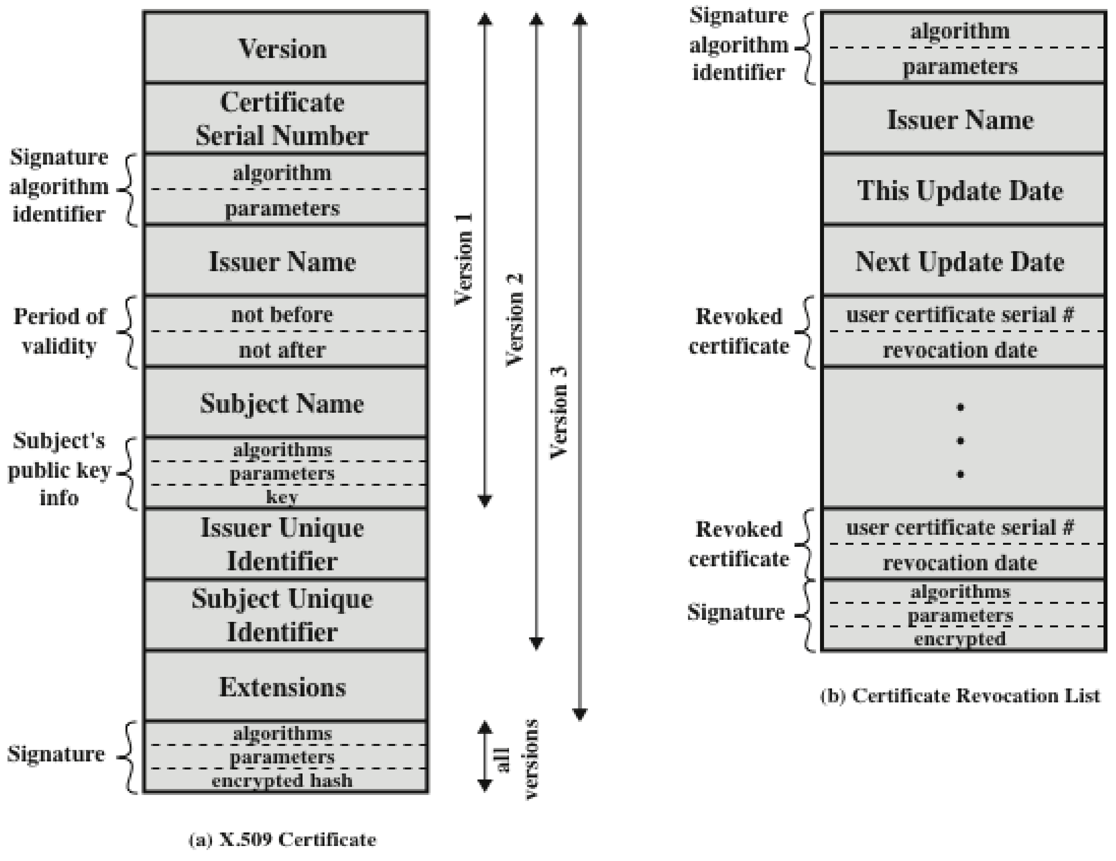


Figure 14.15 X.509 Formats

Anonymous Cash

As the Information Age progresses, network connections are becoming more and more important to our country. Technology based on computers has a big impact on how we access, store, and share information.

Electronic commerce, which involves carrying out financial transactions using digital information transferred over telecommunications networks, is one of the technology's most significant applications. The creation of reliable and secure electronic payment systems is a crucial necessity for electronic commerce. The development of the Internet, which is expected to be a key platform for future electronic commerce, has brought attention to the need for security. Debit cards, credit cards, stored value cards, digital checks, and other types of electronic payment systems are only a few examples. The typical security features for such systems are nonrepudiation (preventing a transaction from being later denied), authenticity (providing user identification and message integrity), and privacy (protection against eavesdropping).

Electronic cash, as the name suggests, is an effort to create an electronic payment system that is modelled after our paper money system. Paper money has the following qualities: it is portable (easily carried), easily accepted (as legal tender), transferrable (without involving the financial network), anonymous (no record of who spent the money), untraceable (no record of where money is spent), and it may be used to make "change." Electronic cash's creators put a lot of effort into keeping the attributes of anonymity and untraceability. As a result, electronic currency is described as an electronic payment system that offers user anonymity and payment untraceability in addition to the security aspects. Digital signatures are typically used in electronic cash systems to accomplish these security objectives. They can be viewed as the handwritten signature's digital equivalent. The foundation of digital signatures is public key cryptography. Each user in such a cryptosystem possesses a secret key as well as a public key. The public key is required to validate a digital signature, whereas the secret key is used to produce them. One must be confident they are aware of the owner of a particular public key to determine who signed the data (also known as the message). The solution to this key management issue necessitates the use of an authentication infrastructure. To protect the confidentiality of the secret keys, the system also needs to have sufficient network and physical security.

From a law enforcement perspective, these schemes are much less effective. Particularly, there is a far greater potential risk of money laundering and counterfeiting than there is with paper money. Any electronic payment system has these issues, but the existence of anonymity makes them significantly worse. In fact, the widespread usage of electronic cash would make the country's financial system more susceptible to attacks from information warfare.

The electronic payment scenario assumes three kinds of players:

- a payer or consumer, whom we will name Alice.
- a payee, such as a merchant. We will name the payee Bob.
- a financial network with whom both Alice and Bob have accounts who we refer to as a bank.

With the rise of telecommunications and the Internet, it is increasingly the case that electronic commerce takes place using a transmission medium not under the control of the financial system. It is therefore necessary to take steps to insure the security of the messages sent along such a medium.

The necessary security properties are:

1. Privacy, or protection against eavesdropping. This is obviously of importance for transactions involving, e.g., credit card numbers sent on the Internet.

2. User identification, or protection against impersonation. Clearly, any scheme for electronic commerce must require that a user knows with whom she is dealing (if only as an alias or credit card number).
3. Message integrity, or protection against tampering or substitution. One must know that the recipient's copy of the message is the same as what was sent.
4. Nonrepudiation, or protection against later denial of a transaction. This is clearly necessary for electronic commerce, for such things as digital receipts and payments.

We now present a simplified electronic cash system, without the anonymity features.

PROTOCOL 1: On-line electronic payment.

Withdrawal:

- Alice sends a withdrawal request to the Bank.
- Bank prepares an electronic coin and digitally signs it.
- Bank sends coin to Alice and debits her account.

Payment/Deposit:

- Alice gives Bob the coin.
- Bob contacts Bank and sends coin.
- Bank verifies the Bank's digital signature.
- Bank verifies that coin has not already been spent.
- Bank consults its withdrawal records to confirm Alice's withdrawal.
- Bank enters coin in spent-coin database.
- Bank credits Bob's account and informs Bob.
- Bob gives Alice the merchandise.

Digital signatures are used in the methods to ensure authenticity. Other methods may have been used to provide the authenticity characteristics, but to support the anonymity mechanisms we are about to implement, digital signatures are required.

We now modify the above protocols to include payment untraceability. For this, it is necessary that the Bank is not able to link a specific withdrawal with a specific deposit. This is accomplished using a special kind of digital signature called a blind signature.

PROTOCOL 2: Untraceable On-line electronic payment.

Withdrawal:

- Alice creates an electronic coin and blinds it.
- Alice sends the blinded coin to the Bank with a withdrawal request.
- Bank digitally signs the blinded coin.
- Bank sends the signed blinded coin to Alice and debits her account.
- Alice unblinds the signed coin.

Payment/Deposit:

- Alice gives Bob the coin.
- Bob contacts Bank and sends coin.
- Bank verifies the Bank's digital signature.
- Bank verifies that coin has not already been spent.
- Bank enters coin in spent-coin database.
- Bank credits Bob's account and informs Bob.
- Bob gives Alice the merchandise

Electronic cash's untraceability makes it difficult to identify cases of tax evasion and money laundering because there is no method to connect the payment and payee. It is conceivable to create a system that includes the option to restore traceability via an escrow mechanism to combat this issue. If certain requirements are met, a deposit or withdrawal record may be sent to a party who is widely regarded as trustworthy and who possesses the key necessary to decrypt the data linking a deposit to a withdrawal or vice versa. By doing so, the payer or payee in a certain transaction will be identified. This does not, however, address the issue of token forging since it might not be possible to identify which contributions are questionable. In conclusion, the potential risks in electronic commerce are magnified when anonymity is present. Anonymity creates the potential for large sums of counterfeit money to go undetected by preventing identification of forged coins. Anonymity also provides an avenue for laundering money and evading taxes that is difficult to combat without resorting to escrow mechanisms. Anonymity can be provided at varying levels but increasing the level of anonymity also increases the potential damages. It is necessary to weigh the need for anonymity with these concerns. It may well be concluded that these problems are best avoided by using a secure electronic payment system that provides privacy, but not anonymity.