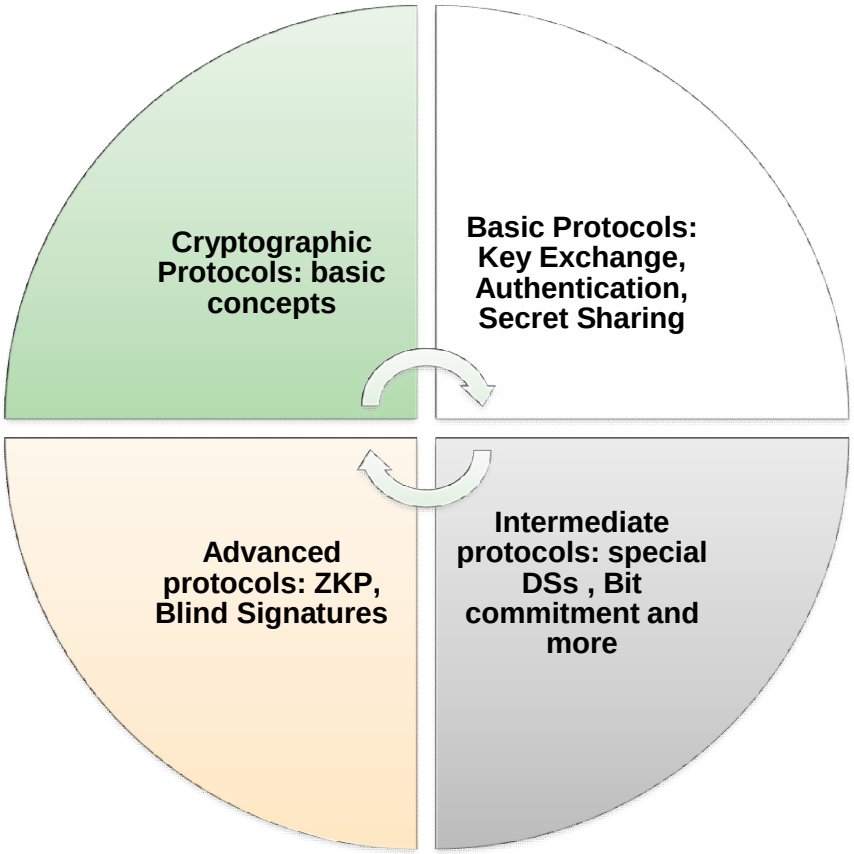
Cryptographic Protocols I

Van Nguyen – HUT

Hanoi – 2010

Agenda


Cryptographic Protocols: basic concepts
Basic Protocols: Key Exchange, Authentication, Secret Sharing
Advanced protocols: ZKP, Blind Signatures
Intermediate protocols: special DSs , Bit commitment and more

Cryptographic Protocol: basic idea

- Cryptography solves problems that involve secrecy, authentication, integrity, and dishonest people.
- A protocol is a series of steps, involving two or more parties, designed to accomplish a task.
  - "series of steps": the protocol has a sequence, from start to finish.
    - Every step must be executed in turn, and no step can be taken before the previous step is finished.
  - "Involving two or more parties": at least two people are required to complete the protocol
    - one person alone does not make a protocol. A person alone can perform a series of steps to accomplish a task, but this is not a protocol.
  - "designed to accomplish a task": the protocol must achieve something.

Cryptographic Protocol: other characteristics

- Everyone involved in the protocol must
  - know the protocol and all of the steps to follow in advance
  - agree to follow it
- The protocol must be unambiguous:
  - each step must be well defined
  - there must be no chance of a misunderstanding
- The protocol must be complete:
  - there must be a specified action for every possible situation

- A cryptographic protocol involves some cryptographic algorithm, but generally the goal of the protocol is something beyond simple secrecy.
- The parties might want
  - to share parts of their secrets to compute a value,
  - jointly generate a random sequence
  - convince one another of their identity
  - or simultaneously sign a contract.
- The whole point of using cryptography in a protocol is to prevent or detect eavesdropping and cheating
  - — It should not be possible to do more or learn more than what is specified in the protocol.

List of regular players

- Alice: First participant in all the protocols
- Bob: Second participant in all the protocols
- Carol: Participant in the three- and four-party protocols
- Dave: Participant in the four-party protocols
- Eve: Eavesdropper
- Mallory: Malicious active attacker
- Trent: Trusted arbitrator
- Walter: Warden who will be guarding Alice and Bob in some protocols
- Peggy: the Prover
- Victor: the Verifier

Arbitrated Protocols

- Arbitrator: disinterested third party trusted to help complete a protocol between two mutually distrustful parties.
- In the real world, lawyers are often used as arbitrators.
  - E.g. Alice is selling a car to Bob, a stranger. Bob wants to pay by check, but Alice has no way of knowing if the check is good.
  - Enter a lawyer trusted by both. With his help, Alice and Bob can use the following protocol to ensure that neither cheats the other:

    (1) Alice gives the title to the lawyer

    (2) Bob gives the check to Alice.

    (3) Alice deposits the check.

    (4) After waiting a specified time period for the check to clear, the lawyer gives the title to Bob. If the check does not clear within the specified time period, Alice shows proof of this to the lawyer and the lawyer returns the title to Alice.

Adjudicated Protocols

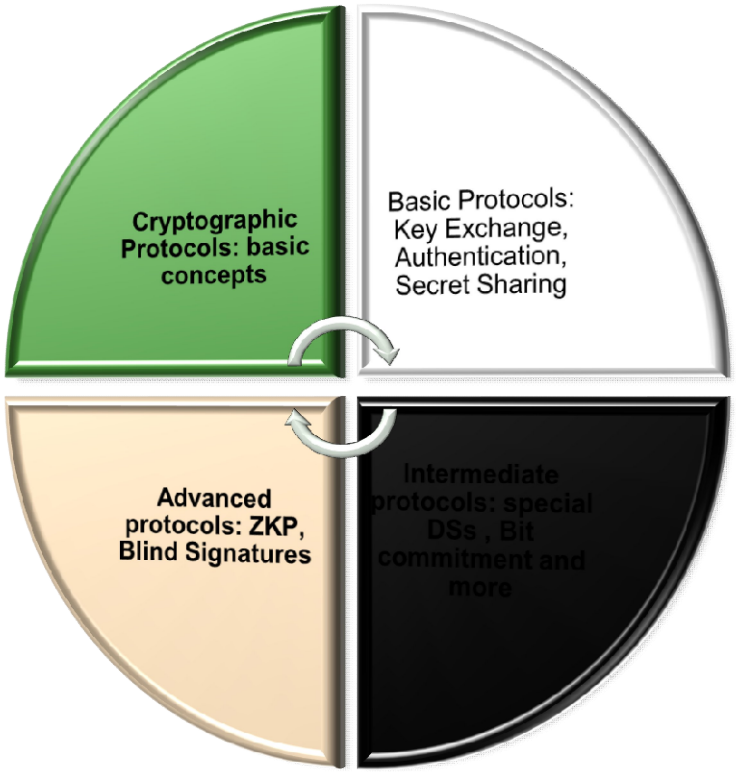
- Because of the high cost of hiring arbitrators, arbitrated protocols can be subdivided into 2 subprotocols
  - Non-arbitrated subprotocol, executed every time parties want to complete the protocol.
  - an arbitrated subprotocol, executed only in exceptional circumstances— when there is a dispute.

Adjudicated Protocols

- Example: The contract-signing protocol can be formalized in this way:
  - Nonarbitrated subprotocol (executed every time):
    - (1) Alice and Bob negotiate the terms of the contract.
    - (2) Alice signs the contract.
    - (3) Bob signs the contract.
  - Adjudicated subprotocol (executed only in case of a dispute):
    - (4) Alice and Bob appear before a judge.
    - (5) Alice presents her evidence.
    - (6) Bob presents his evidence.
    - (7) The judge rules on the evidence.

Self-Enforcing Protocols

- A self-enforcing protocol is the best type of protocol: The protocol itself guarantees fairness.
  - No arbitrator is required to complete the protocol.
  - No adjudicator is required to resolve disputes.
- The protocol is constructed so that there cannot be any disputes
  - If one of the parties tries to cheat, the other party immediately detects the cheating and the protocol stops.
- Unfortunately, there is not a self-enforcing protocol for every situation


Cryptographic Protocols: basic concepts
Basic Protocols: Key Exchange, Authentication, Secret Sharing
Advanced protocols: ZKP, Blind Signatures
Intermediate protocols: special DSs , Bit commitment and more

Key Exchange with Symmetric Cryptography

- Assume that Alice and Bob each share a secret key with the Key Distribution Center (KDC)—Trent in our protocols.
  - These keys must be in place before the start of the protocol

(1) Alice calls Trent and requests a session key to communicate with Bob.

(2) Trent generates a random session key. He encrypts two copies of it: one in Alice's key and the other in Bob's key. Trent sends both copies to Alice.

(3) Alice decrypts her copy of the session key.

(4) Alice sends Bob his copy of the session key.

(5) Bob decrypts his copy of the session key.

(6) Both Alice and Bob use this session key to communicate securely.

Key Exchange with Public-Key Cryptography

- Alice and Bob use public-key cryptography to agree on a session key, and use that session key to encrypt data.
- In some practical implementations, both Alice's and Bob's signed public keys will be available on a database.

(1) Alice gets Bob's public key from the KDC.

(2) Alice generates a random session key, encrypts it using Bob's public key, and sends it to Bob.

(3) Bob then decrypts Alice's message using his private key.

(4) Both of them encrypt their communications using the same session key.

Man-in-the-Middle Attack

- Mallory is a lot more powerful than Eve
  - can also modify messages, delete messages, and generate totally new ones
- Mallory can imitate Bob when talking to Alice and imitate Alice when talking to Bob

(1) Alice sends Bob her pk. Mallory intercepts this key and sends Bob his own pk.

(2) Bob sends Alice his pk. Mallory intercepts, sends Alice his own pk.

(3) When Alice sends a message to Bob, encrypted in “Bob’s” pk, Mallory intercepts it:

Since the message is really encrypted with his own public key, he decrypts it with his private key, re-encrypts it with Bob’s public key, and sends it on to Bob.

(4) When Bob sends a message to Alice, encrypted in “Alice’s” pk, Mallory intercepts it: he decrypts it with his private key, re-encrypts it with Alice’s public key, and sends it on to Alice.

- The interlock protocol, invented by Ron Rivest and Adi Shamir can protect against the man-in-the-middle attack.

  (1) Alice sends Bob her public key.

  (2) Bob sends Alice his public key.

  (3) Alice encrypts her message using Bob's public key. She sends half of the encrypted message to Bob.

  (4) Bob encrypts his message using Alice's public key. He sends half of the encrypted message to Alice.

  (5) Alice sends the other half of her encrypted message to Bob.

  (6) Bob puts the two halves of Alice's message together and decrypts it with his private key. Bob sends the other half of his encrypted message to Alice.

  (7) Alice puts the two halves of Bob's message together and decrypts it with her private key.

Authentication

- When Alice logs into a host computer (or an ATM, …), how does the host know who she is? How does the host know she is not Eve trying to falsify Alice's identity?
  - Traditionally, passwords solve this problem. Alice enters her password, and the host confirms that it is correct.
  - Both Alice and the host know this secret piece of knowledge and the host requests it from Alice every time she tries to log in.
- *Authentication Using One-Way Functions*
  - the host does not need to know the passwords:just has to be able to differentiate valid passwords from invalid passwords.
  - (1) Alice sends the host her password.
  - (2) The host performs a one-way function on the password.
  - (3) The host compares the result of the one-way function to the value it previously stored.

Authentication

- *Dictionary Attacks and Salt*
- *SKEY*
- *Authentication Using Public-Key Cryptography*
- *Mutual Authentication Using the Interlock Protocol*
- *SKID*

Authentication and Key Exchange

- Needham-Schroeder

(1) Alice sends a message to Trent consisting of her name, Bob's name, and a random number: A,B,R_A

(2) Trent generates a random session key. He encrypts a message consisting of a random session key and Alice's name with the secret key he shares with Bob. Then he encrypts Alice's random value, Bob's name, the key, and the encrypted message with the secret key he shares with Alice, and sends her the encrypted:

$$E_A(R_A,B,K,E_B(K,A))$$

(3) Alice decrypts the message and extracts *K. She confirms that RA is the same value* that she sent Trent in step (1). Then she sends Bob the message that Trent encrypted in his key: $E_B(K,A)$

(4) Bob decrypts the message and extracts *K. He then generates another random value,* R_B*. He encrypts the message with K and sends it to Alice.* $E_K(R_B)$

(5) Alice decrypts the message with *K. She generates RB - 1 and encrypts it with K. Then* she sends the message back to Bob: $E_B(R_B - 1)$

Authentication and Key Exchange

- Otway-Rees:
  - Fix the replay attack to Needham-Schroeder's
- Kerberos
- DASS
- Denning-Sacco

Secret Spliting

- There are ways to take a message and divide it up into pieces
  - Each piece by itself means nothing, but put them together and the message appears.
- The simplest sharing scheme splits a message between two people:

(1) Trent generates a random-bit string, *R, the same length as the message, M.*

(2) Trent XORs *M with R to generate S:* $M \oplus R = S$

(3) Trent gives *R to Alice and S to Bob*

To reconstruct the message, Alice and Bob have only one step to do:

(4) Alice and Bob XOR their pieces together to reconstruct the message:

- $R \oplus S = M$

Secret Sharing

- A threshold scheme can take any message (a secret recipe, launch codes etc.) and divide it into *n pieces, called shadows or shares, such that any m of them can be* used to reconstruct the message. More precisely, this is called an *(m,n)-threshold scheme.*
  - With a (3,4)-threshold scheme, Trent can divide his secret sauce recipe among Alice, Bob, Carol, and Dave, such that any three of them can put their shadows together and reconstruct the message.
  - If Carol is on vacation, Alice, Bob, and Dave can do it. If Bob gets run over by a bus, Alice, Carol, and Dave can do it. However, if Bob gets run over by a bus while Carol is on vacation, Alice and Dave can't reconstruct the message by themselves.

Special Digital Signatures

- Undeniable signatures
- Designated confirmer signatures
- Proxy Signatures
- Group Signatures
- Fail-Stop Digital Signatures

Bit Commitment

- Using symmetric cryptography:

(1) Bob generates a random-bit string, *R, and sends it to Alice*

(2) Alice creates a message consisting of the bit she wishes to commit to, *b (it can actually* be several bits), and Bob's random string. She encrypts it with some random key, *K, and sends* to Bob: $E_K(R,b)$

That is the commitment portion of the protocol. Bob cannot decrypt the message, so he does not know what the bit is.

- When it comes time for Alice to reveal her bit, the protocol continues:

  (3) Alice sends Bob the key.

  (4) Bob decrypts the message to reveal the bit. He checks his random string to verify the bit's validity

Bit Commitment

- Using one-way functions:

  (1) Alice generates two random-bit strings, *R1 and R2.*

  (2) Alice creates a message consisting of her random strings and the bit she wishes to commit to (it can actually be several bits).Then computes the one-way function on the message and sends the result, as well as one of the random strings, to Bob: $H(R_1,R_2,b),R_1$

- When it comes time for Alice to reveal her bit, the protocol continues:

  (4) Alice sends Bob the original message: (*R1,R2,b)*

  (5) Bob computes the one-way function on the message and compares it and *R1, with the* value and random string he received in step (3). If they match, the bit is valid.

Fair Coin Flips

- Can use Bit Commitment Protocol
- *Coin Flipping Using One-Way Functions*

  Assume Alice and Bob agree on a one-way function *f*:

  (1) Alice chooses a random number, *x. She computes y = f(x),*

  (2) Alice sends *y to Bob.*

  (3) Bob guesses whether *x is even or odd and sends his guess to Alice.*

  (4) If Bob's guess is correct, the result of the coin flip is heads. If Bob's guess is incorrect, the result of the coin flip is tails. Alice announces the result of the coin flip and sends *x to Bob.*

  (5) Bob confirms that *y = f(x).*

Zero-Knowledge Proofs

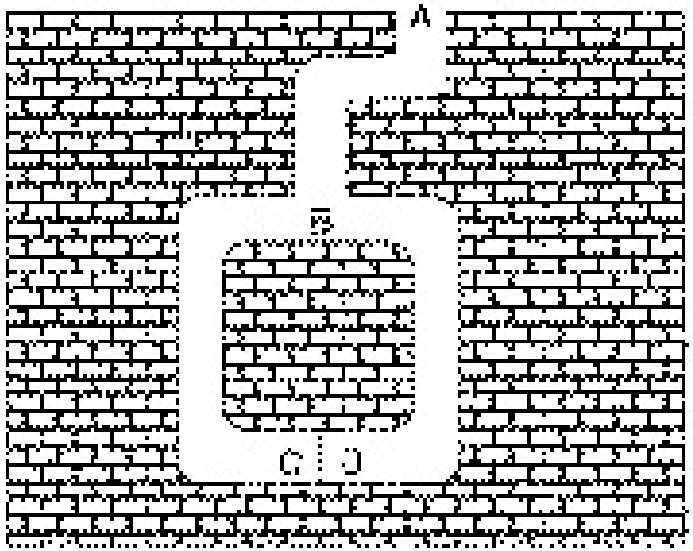
- Alice: “I know the password to the Federal Reserve System computer, the ingredients in McDonald’s secret sauce, and the contents of Volume 4 of Knuth.”
- Bob: “No, you don’t.”
- Alice: “Yes, I do.”
- Bob: “Do not!”
- Alice: “Do too!”
- Bob: “Prove it!”
- Alice: “All right. I’ll tell you.” She whispers in Bob’s ear.
- Bob: “That’s interesting. Now I know it, too. I’m going to tell *The Washington Post.”*
- Alice: “Oops.”

Zero-Knowledge Proofs

- Using one-way functions, Peggy could perform a zero-knowledge proof.
  - This protocol proves to Victor that Peggy does have a piece of information BUT
  - It does not give Victor any way to know what the information is.
- These proofs take the form of interactive protocols. Victor asks Peggy a series of questions.
  - If Peggy knows the secret, she can answer all the questions correctly.
    - After 10 or so questions ➔ Victor will be convinced.
  - If she does not, she has some chance—50 percent chance
  - Yet none of the questions or answers gives Victor any information about Peggy's information—only about her knowledge of it.

Zero-Knowledge Proofs

- The cave has a secret.
  - Someone who knows the magic words can open the secret door between C and D.
  - To everyone else, both passages lead to dead ends.
- Peggy knows the secret of the cave.
  - She wants to prove her knowledge to Victor, but she doesn't want to reveal the magic words.



Blind Signatures

- An essential feature of digital signature protocols is that the signer knows what he is signing.
- We might want people to sign documents without ever seeing their contents.
  - Bob is a notary public. Alice wants him to sign a document, but does not want him to have any idea what he is signing.
    - Bob doesn't care what the document says; he is just certifying that he notarized it at a certain time. He is willing to go along with this.
  - (1) Alice takes the document and multiplies it by a random value. This random value is called a blinding factor.
  - (2) Alice sends the blinded document to Bob.
  - (3) Bob signs the blinded document.
  - (4) Alice divides out the blinding factor, leaving the original document signed by Bob.

Blind Signatures

- The properties of completely blind signatures are:

1. Bob's signature on the document is valid.
   - The signature is a proof that Bob signed the document.
   - It will convince Bob that he signed the document if it is ever shown to him.
   - It also has all of the other properties of digital signatures.
2. Bob cannot correlate the signed document with the act of signing the document.
   - Even if he keeps records of every blind signature he makes, he cannot determine when he signed any given document.