
CRYPTOGRAPHIC PROTOCOLS

CIS 400/628 — Spring 2005
Introduction to Cryptography

This is based on Chapters 14 and 15 of **“Practical Cryptography”**
by Niels Ferguson and Bruce Schneier

THE SECURE CHANNEL, I

A secure channel a secure connection between two parties

A basic tool in crypto protocols.

Roles Alice, Bob, Eve

Key Alice and Bob share a secret key K .

Requirements

- ▶ K is known only to Alice and Bob.
- ▶ Every time the secure channel is initialized a new value is generated for K .

Messages A sequence of discrete messages.

Assumption:

- ▶ The transport layer is **not** reliable.
E.g., TCP/IP is reliable against random events, but not an active and clever attacker.

More ...

THE SECURE CHANNEL, II

Security Properties

Secrecy

Eve does not learn anything about the messages sent except their timing and size.

No Junk

If Alice sends $m_1, m_2 \dots$, Bob receives a subsequence m'_1, m'_2, \dots
I.e., dropping some (maybe none) messages from $m_1, m_2 \dots$ produces m'_1, m'_2, \dots

Bob can always request a resend of missing info.

Authentication & encryption Which first?

For details: See Chapter 8 of FS, but ...

THE SECURE CHANNEL OUTLINE

Message Numbering

- ▶ Why?
- ▶ Number 1 through $2^{32} - 1$. After $2^{32} - 1$ messages, refresh the key.

Authentication

- ▶ Use a standard MAC: **HMAC-SHA-256**.
- ▶ For message i :

$$a_i := \text{MAC}(i \parallel \ell(x_i) \parallel x_i \parallel m_i)$$

x_i = context data for Bob.

$\ell(x_i)$ = the length in bytes of x_i

Encryption

- ▶ Use AES in CTR mode.
- ▶ Messages no larger than $16 \cdot 2^{32}$ bytes.

Frame Format

i coded as a 32-bit integer, least significant byte first followed by the encrypted m_i and a_i .

PROTOCOLS

- ▶ A **protocol** is a scripted exchange of messages in which various parties play particular roles.
- ▶ Designing and implementing these in real life is tricky.
- ▶ The cryptography is hard, but it is the easy part.
- ▶ Putting the pieces together in a way that preserves security is **really** hard.

ROLES

- ▶ There are roles (Alice and Bob) and there are individuals (you, amazon.com, ...)
- ▶ An individual can play different roles at different times.
- ▶ An individual can play multiple roles in a protocol. (E.g., man-in-the-middle attacks)
- ▶ An individual playing a role typically has no idea who is playing the other roles and what their motivations are.
- ▶ The protocol has to be strong enough for that individual to trust the outcome even under a completely paranoid view of the other parties.

TRUST

- ▶ No trust \implies no transactions.
- ▶ Sources of trust
 - Ethics
 - Reputation
 - Law
 - Physical Threat
 - M.A.D.
- ▶ Trust is hard to develop over the Internet
- ▶ Trust is a matter of degree
- ▶ Risk is the converse of trust
 - Businesses think in terms of risks.

INCENTIVES

- ▶ Incentive structures and crypto protocols interact
 - The protocols depend on incentive structures.
E.g., They cannot stop a merchant from cheating a customer, **but** if the protocol gives the customer a proof of purchase . . .
 - The protocols change incentive structures.
 - ★ They make certain things impossible.
(E.g., certain types of forgery)
 - ★ They make new things possible.
(E.g., online banking, and hence, online break-ins to steal money.)

PRACTICAL PARANOIA

- ▶ The function of a crypto protocol is to minimize the amount of trust required in a transaction.

This involves

- minimizing the number of agents involved
 - minimizing the need of these agents to trust each other
- ▶ **The paranoia model**
= all other agents are out to cheat you—perhaps in collusion
 - ▶ In a protocol, any point of deviation from this model **must** be documented. **(The SSL example)**
 - ▶ Each point of required trust implies a risk to be dealt with.

LAYERS OF PROTOCOLS

the high level description
protocol execution states
message encoding and parsing
protocol and message ids
the transport layer

The Transport layer

- ▶ E.g., TCP/IP
- ▶ Should be able to send arbitrary sized messages
- ▶ A secure channel makes everything easier, but we don't always have one.

LAYERS OF PROTOCOLS, II

Protocol and Message IDs

- ▶ A message needs to contain:
 - what protocol it belongs to and
 - what part of the protocol it is.
- ▶ These are not secure.
- ▶ They help detect and protect against accidental errors.

Message Encoding and Parsing

- ▶ The encoding has to be flexible enough to handle variable length messages.
- ▶ Fixed length fields are simple, but they seldom survive version changes.

Protocol Execution States

- ▶ Event-driven. A state-machine is a good idea.

PROTOCOL PLUMBING: ERRORS

Dealing with errors

- ▶ Given the dirty environment they are operating in, there are **many** possible errors in a protocol.
- ▶ However, errors are a potential way to attack a system.
- ▶ Give as little away as possible as to what the problem was. In a cypto context **“There was an error.”** is info enough.
- ▶ The time it takes to detect and reply to an error can also give away information. **Timing attacks**
- ▶ **The smart card PIN example.**

PROTOCOL PLUMBING: REPLAYS AND RETRIES

A replay attack Eve records a message and later sends that message again.

A retry Alice didn't get an acknowledgement from Bob about the last messages, so she sends it again.

How is Bob to securely handle repeated messages?

Upon receiving a message:

- ▶ If it is the one expected, do the normal things.
- ▶ If it is a message “from the future” (e.g., expecting #8, receiving #23), ignore it.
- ▶ If it is a message “from the past”:
 - If it is identical to one previous received, then send the **identical** reply.
 - If it has the same message ID as a previous one, but has different contents, treat as a protocol error.
 - If it is a really old message, the safe thing is to ignore it.

A DH+ PROTOCOL

Goal

Alice and Bob want to agree on a secret session key for a secure channel.

Assumption

We assume A&B can authenticate messages to one another.
A&B may have a shared secret key for authentication.

Question:

If they already share a secret key, why bother with another?

Answer:

- ▶ Compartmentalization, just like in spy networks.
- ▶ If the session key is compromised, the long-term key is safe.
- ▶ If the long-term key is compromised and Eve hasn't recorded yesterday's key negotiation, then yesterday's exchanges are safe.

A FIRST TRY

Setup

Alice and Bob know (p, q, g) , where $p = 2q + 1$ and q are primes, $g \neq 1$, but $g^q \equiv 1 \pmod{p}$.

Alice

Chooses $x \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$ and sends $X = g^x \pmod{p}$ to Bob.

Bob

Chooses $y \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$, sends $Y = g^y \pmod{p}$ to Alice, and computes $k = X^y \pmod{p}$.

Alice

Computes $k = Y^x \pmod{p}$ and sends $\text{Auth}_A(k)$ to Bob.

Bob

Sends $\text{Auth}_B(k)$ to Alice.

Henceforth we drop the \pmod{p} 's

PROBLEMS WITH THE FIRST TRY

- ▶ Using constants for (p, q, g) is a bad idea.
- ▶ It can be done with fewer messages.
- ▶ Using k in the authentication messages is a bad idea.
Use one secret for one purpose only.
E.g., If the authentication method is weak, a few bits of k could leak.
- ▶ The authentication messages are too similar.
E.g., If $\text{Auth}_X(\cdot)$ is based on a symmetric cyrptosystem, then $\text{Auth}_A(k) = \text{Auth}_B(k)$.
Why should Alice beleive the message is really from Bob?
- ▶ We have to be sure that k isn't used until the authentications check out.

NOTES

Protocols last forever

- ▶ So they need to be designed to be future proof.
- ▶ E.g., that is why having (p, q, g) is a bad idea.
- ▶ A protocol switch/rollback is a point of attack.

Authentication conventions

- ▶ At any point in the protocol, the authentication data consists of all data exchanged so far.
E.g., In try 1, Alice's authentication data we consist of: X and Y and Bob's would be X, Y , and $\text{Auth}_A(\dots)$.
- ▶ This is reasonably cheap and stops lots of attacks.
- ▶ Henceforth:
 Auth_A = Alice's authentication message at this point of the protocol.
 Auth_B = Bob's authentication message at this point of the protocol.

A SECOND TRY

Alice

Chooses (p, q, g) and $x \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$ and computes $X = g^x$.
Sends $(p, q, g), X, \text{Auth}_A$ to Bob.

Bob

Checks $(p, q, g), X, \text{Auth}_A$, chooses $y \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$, and computes $Y = g^y$.
Sends Y, Auth_B to Alice. Computes $k = X^y$.

Alice

Checks Y, Auth_B . Computes $k = Y^x$.

PROBLEMS

- ▶ What if Bob is fussier than Alice and wants bigger primes?
- ▶ How can Bob tell his is really talking to Alice and not a replay ghost?

A THIRD TRY

Alice

Chooses $s := \min$ size of p and $N_A \stackrel{\text{ran}}{\in} \{0, \dots, 2^{256} - 1\}$.

Sends s, N_A to Bob.

N_A is a nonce.

Bob

Chooses (p, q, g) and $x \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$. Computes $X = g^x$.

Sends $(p, q, g), X, \text{Auth}_B$ to Alice.

Alice

Checks $(p, q, g), X, \text{Auth}_B$. Chooses $y \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$. Computes $Y = g^y$.

Sends Y, Auth_A to Bob. Computes $k = X^y$.

Bob

Checks Y, Auth_A and computes $k = Y^x$.

PROBLEMS

- ▶ k is of variable size.
- ▶ k has algebraic structure.

FOURTH TRY

Alice

Chooses $s := \min$ size of p and $N_A \stackrel{\text{ran}}{\in} \{0, \dots, 2^{256} - 1\}$.

Sends s, N_A to Bob.

N_A is a nonce.

Bob

Chooses (p, q, g) and $x \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$. Computes $X = g^x$.

Sends $(p, q, g), X, \text{Auth}_B$ to Alice.

Alice

Checks $(p, q, g), X, \text{Auth}_B$. Chooses $y \stackrel{\text{ran}}{\in} \mathbb{Z}_q^*$. Computes $Y = g^y$.

Sends Y, Auth_A to Bob.

Computes $k = \text{SHA}_{d-256}(X^y)$.

Bob

Checks Y, Auth_A .

Computes $k = \text{SHA}_{d-256}(Y^x)$.

See page 270 of F&S for the long form.

ALICE'S VIEW OF THE PROTOCOL

- ▶ She receives only one message from Bob.
- ▶ Since this message contains her nonce, it can't come from a replay ghost.
∴ it must really be Bob.
- ▶ Since the DH parameters check out, when she chooses y and ships out Y , she knows that Bob is the only one who can construct the key.
∴ So the basic DH assumptions hold for Alice.

BOB'S VIEW OF THE PROTOCOL

- ▶ The first message has no useful info.
- ▶ The third message has Alice's authentication and X , which Bob generated.
∴ It must be Alice, not a replay ghost.
- ▶ Since Bob generated the DH parameters, he knows they are OK.
- ▶ Since Alice authenticated her Y , Bob knows that Alice is the only person other person who could compute the key.
∴ So the basic DH assumptions hold for Alice.

EVE'S VIEW OF THE PROTOCOL

- ▶ A passive attack is useless,
So long as the DH parameters are safe.
 - ▶ The two authentications stop us changing data elements.
 - ▶ The nonce N_A and X stop replay attacks.
 - ▶ Eve could increase s , but that does her no good.
-
- ▶ Unauthenticated data elements
give an attacker things to play with.
 - ▶ Authenticating all data:
paranoid, but provides defense in depth.

KEY COMPROMISE

Alice loses her authentication key (without the knowledge of Eve)

She can't run the protocol, but can use already established session keys.

If Alice loses her session key (without the knowledge of Eve)

She reruns the protocol to get a new session key.

Eve steals Alice's authentication key

Eve can impersonate Alice until Bob finds out the problem.

However, previous Alice-Bob communications are still secure.

Forward secrecy

Eve steals Alice's session key k

k provides no info about other session keys

or about Alice and Bob's authentication keys.