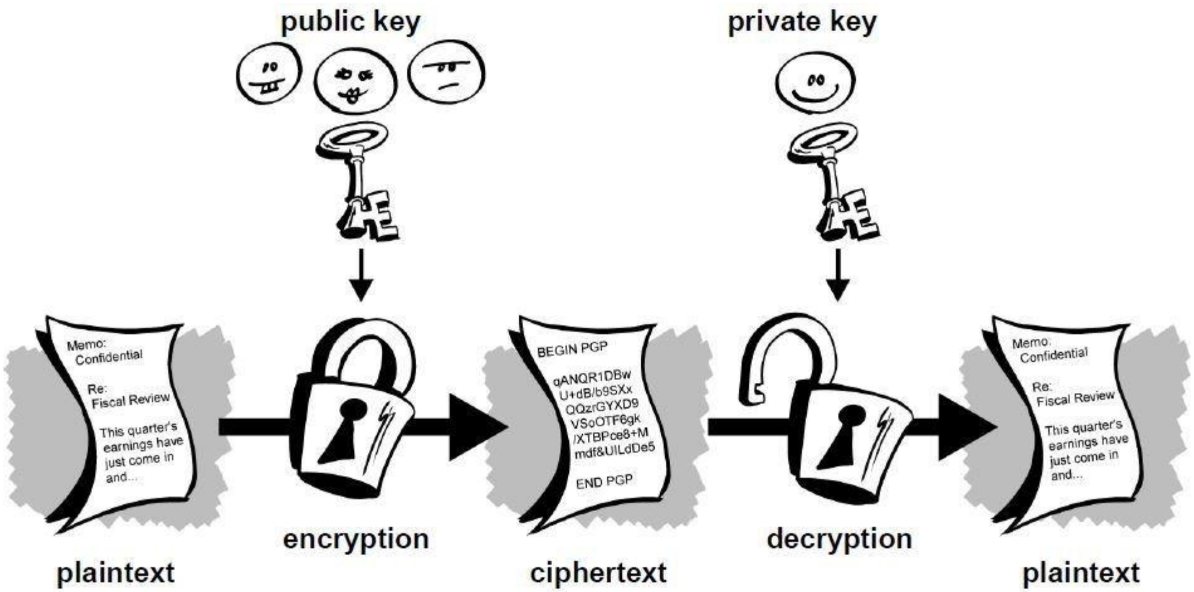
Introduction to Cryptography

By

Vipul Goyal



Hard Problems in Cryptography

Recap: Hard Problem 1: Discrete Log Problem

Discrete Log problem (DLP):

given g, N and $g^x \bmod N$, output x

DLP considered to hard (for carefully chosen g, N)

We will see how use DLP to build:

- Public-key encryption
- Private-key encryption (with reusable short key)
- Digital Signatures

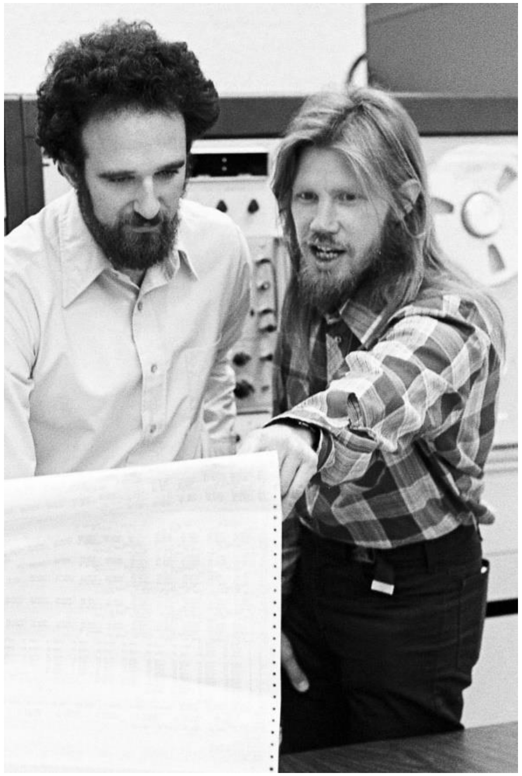
Going forward: all arithmetic will be mod N.

Will not write mod N explicitly

HP2: Computational Diffie-Hellman (CDH) Problem

Given (suitably chosen) g,

$A = g^a$, and $B = g^b$

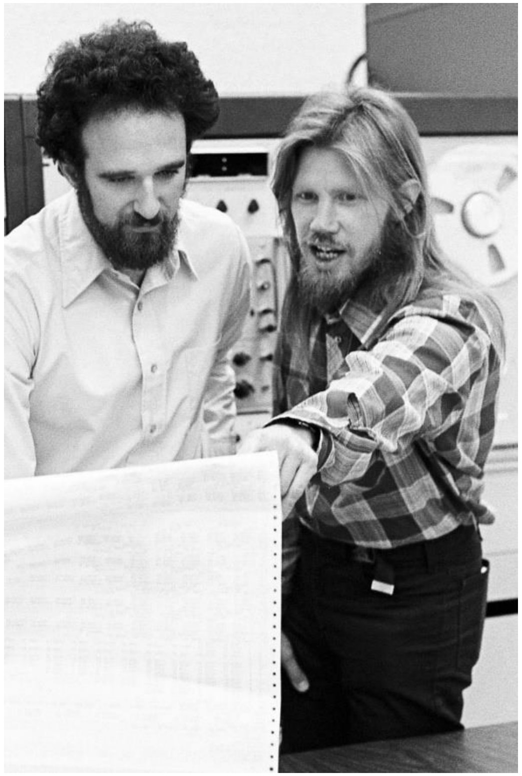


Find C, s.t. (such that)

$C = g^{ab}$

- Note: $A.B = g^a.g^b = g^{a+b}$
- Most natural way of solving CDH:
  - Step1: Find a from g^a
  - Step2: Compute $(g^b)^a = g^{ab}$
  - However Step1 is a hard problem (might be other ways)

HP3: Decisional Diffie-Hellman (DDH) Problem



Given (suitably chosen) g,

$A = g^a$, and $B = g^b$

And either

(1) $C = g^{ab}$, or,

(2) $R = g^r$ (for random r)

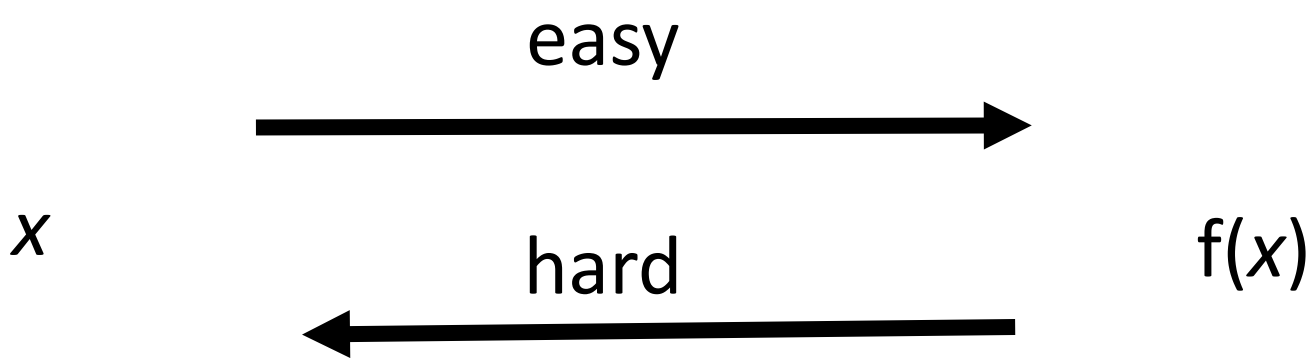
Tell whether its (1) or (2)

Still hard with better than ½

(even given the answer, hard to distinguish it from random

Building on Hard Problems: One-Way Functions (or One-Way Hash Functions)

One-Way Functions (OWF)



Sample random x, compute y = f(x)   (x called pre-image of y)

Given y, hard for any adversary to compute x

In fact, hard to compute any x' s.t. y = f(x')

(hard to compute any valid pre-image)

(will see an application of this to password storage)

Building a OWF?

Given y, hard for any adversary to compute any x' s.t. y = f(x')

(hard to compute any valid pre-image)

Attempt1: $f(x) = x$

Easy to invert, given output, can find input

Attempt2: $f(x) = 0$ (or some other constant)

Every string x' is a valid pre-image because $f(x') = f(x) = 0$

Hence, easy to invert

Attempt3: $f(x) = 2x \bmod N$

To recover x, simply compute 2^{-1} and multiply

OWF based on Discrete Log Assumption

Define OWF f as:

$f(x) = g^x \bmod N$

(description of g, N is public)

DLA: Given $f(x)$, hard to compute x

Other OWF: SHA-256, SHA-1, MD-5.

More complex to understand. Have additional properties.

Applications of OWF: Storing Passwords

Client

Server

p

p

p

Problem: Adv hacks into server. Can steal millions of passwords.

Solution:

1) Server only stores f(p)
2) Client still sends p, server computes f(p) and matches
3) Even if adv learns f(p) by hacking, computing p is hard

Applications of OWF: Storing Passwords

Client

p

p

Server

f(p)

- This solution is used everywhere on the Internet today! Servers should not store your password, only a hash of it.
- That’s why if you forget password: server can’t give it back to you. You can reset instead.

Offline Dictionary Attack on Passwords

Client

Server

p

p

f(p)

- If adv obtains f(p), it can guess millions of different passwords, hash them and check if they match f(p). If match, adv wins!
- Typically: adv checks words from dictionary and common patterns
- Hence: your password should have special characters.

Back to OWF Definition

Client

p

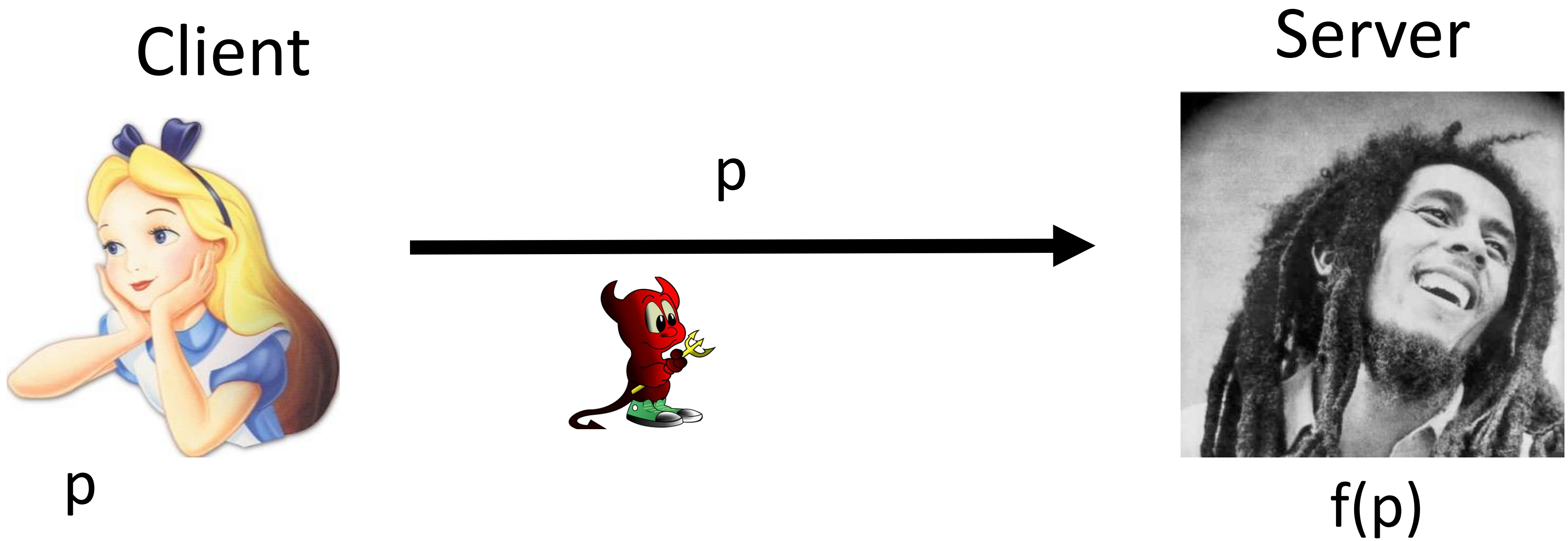
p

Server

f(p)

- Given f(p), say adv can compute p' s.t. f(p') = f(p). Server will also accept p' as valid. Hence, adv still wins!
- That why in OWF: given f(x), finding any x' s.t. f(x') = f(x) must be hard.

Passwords Over the Internet?



Problem: Adv watching the network?

Solution:

1) Use HTTPS to open a secure connection first

   (Client encrypts all messages under Server’s public key)

2) Later: will see how HTTPS works!

Applications of OWF: Digital Signatures

Digital Signatures

VK

M, σ

SK M

sign

σ (signature)

VK, M, σ

verify

0/1

Note: we don't care about hiding M

Defining Digital Signatures

1) Gen: Takes no input, outputs VK and SK
2) Sign: Takes input *SK* and *M*. Outputs σ.
3) Verify: Takes input (M, σ, VK). Outputs 0/1.

Correctness: If (VK, SK) are output of Gen, must have

$$\mathrm{Verify}(M, \mathrm{Sign}(M, SK), VK) = 1$$

Security

Adv is even given:

1) Verification key VK
2) Signatures (σ_1, σ_2,…, σ_q) on messages (M_1, M_2,…, M_q) chosen by him

Adv still can't output a valid signature on a new message

(That is, can't output (σ, M) s.t. Verify(M, σ, VK) = 1 and M is different from all M_i)

One-Time (Digital) Signatures

- Easier to design: we need a scheme which is secure only for signing a single message

Adv is given:

1) Verification key VK
2) Signature σ_1 on any one message M_1 of his choice

Adv can't output a valid signature on a new message

(That is, can't output (σ, M) s.t. $\text{Verify}(M, \sigma, VK) = 1$ and M is different from M_1)

One-Time Signatures: Intuition

- Only two possible messages. Hence only two possible signatures. These are random strings (x_0, x_1). They are generated at random and part of secret key SK.
  - If message is 0, signature is x_0
  - If message is 1, signature is x_1

- How to verify signature? Can't make it a part of the verification key
  - Use an idea similar to storing passwords
  - VK has $f(x_0)$ and $f(x_1)$
  - VK can be used for verification of signature, not for computation

One-Time Signatures

Message length: 1 bit

Gen: pick random (x_0, x_1)

$SK = (x_0, x_1)$

$VK = (f(x_0), f(x_1))$

sanity check

Sign: If $m = 0$, $\sigma = x_0$

If $m = 1$, $\sigma = x_1$

Verify: compute $f(\sigma)$

If $m = 0$, match with $f(x_0)$

If $m = 1$, match with $f(x_1)$

Security

Adv given: VK, and signature on $m = 0$

Wants to compute signature on $m = 1$

Given: $f(x_0)$, $f(x_1)$, x_0

Compute x_1

- x_1 is unrelated to x_0. Hence, $f(x_0)$ and x_0 are not relevant for computing x_1
- Thus, given $f(x_1)$, adv needs to compute x_1
- Needs to invert OWF (hard)!

One-Time Signatures: longer messages

Message length: n bit

For all i, $1 \leq i \leq n$

Gen: pick random (x_0[i], x_1[i]) ([i] is the i-th number picked)

SK = (x_0[i], x_1[i])

VK = (f(x_0[i]), f(x_1[i]))

Sign: If m[i] = 0, include x_0[i] in σ as σ[i] (m[i] = i-th bit of m)

If m[i] = 1, include x_1[i] in σ as σ[i]

Verify: compute f(σ[i])

If m[i] = 0, match with f(x_0[i])

If m[i] = 1, match with f(x_1[i])

Security: say m_1 and m differ in i-th bit

Adv has x_0[i], needs to compute x_1[i]

Needs to invert OWF

Digital Signature Schemes

What about:

- Signing multiple messages with the same key?
- *Can key size be shorter than message?*

Answer: Yes! We will see RSA signatures later

Digital Signature Applications

- Signing documents/forms digitally: Adobe PDF and others

- Online Contract Signing: Two parties can sign a contract over internet

- Digital Degrees and Marksheets

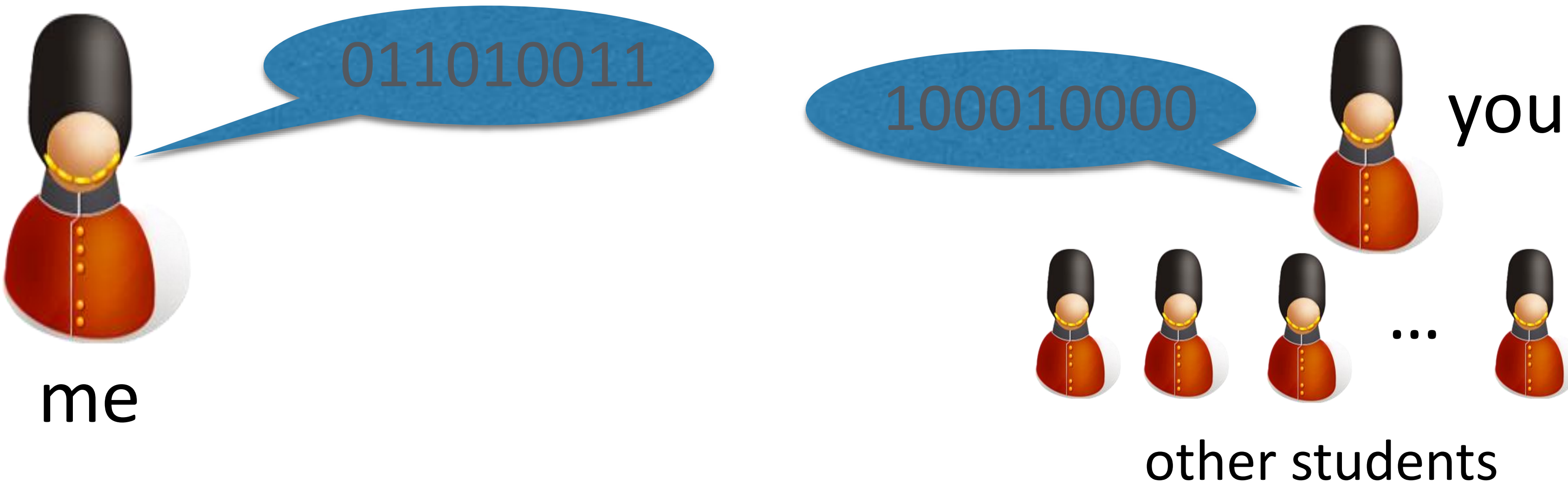
- Bitcoin/Cryptocurrencies and Smart Contracts

Key Exchange over the Internet

Key Exchange

Private key encryption relies on parties having a shared secret

Say I want to communicate with one of you securely. Never met. No private chat. Only speaking publicly on Zoom.



- You and I talk publicly
- You and I now have a shared key
- Other students listening can't compute it
- Impossible?

Key Exchange

Alice

Bob

Can Alice and Bob agree on a secret via a *completely public conversation*?

- Over the internet with adversary watching?

Diffie-Hellman Key Exchange

Alice

Bob

$(N, g), A = g^a$ →

← $B = g^b$

sample a

sample b

Key $K = g^{ab} = A^b = B^a$

Alice: has a and $B = g^b$. Computes $K = B^a = g^{ab}$

Bob: has b and $A = g^a$. Computes $K = A^b = g^{ab}$

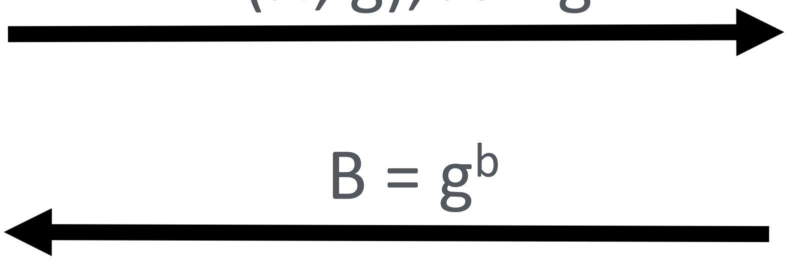
Diffie-Hellman Key Exchange

Alice

Bob

(N, g), A = g^a →

← B = g^b



sample a

sample b

Can Adv compute g^{ab}? Adv only has g^a and g^b

(but neither a nor b)

CDH/DDH say: this is hard!

After Key Exchange

Alice

Bob

$(N, g), A = g^a$ →

← $B = g^b$

← $C = M.g^{ab}$

sample a

sample b

To decrypt: compute $K = g^{ab}$, and K^{-1}

Recover message as $C.K^{-1}$

ElGamal Public-Key Encryption

Defining PKE

1) Gen: No input. Outputs PK and SK
2) Enc: Takes input *PK* and *M*. Outputs *C*.
3) Dec: Takes input *C* and SK. Outputs M.

Correctness: If (PK, SK) are output of Gen, must have

$$\mathrm{Dec}(\mathrm{Enc}(M, PK), SK) = M$$

Security: Should hide the message?

Security

Given C, probability of computing M is very small?

Given C, probability of computing M is at most ½?

Intuition: C should give “no information” about M

Security: Adv can’t tell apart encryption of M from encryption of a random message

Attempt at Building PKE?

Alice, who has never spoken to Bob, wants to send him message m in encrypted form Enc(m)

Recovering m from Enc(m) should be a hard problem

How about Enc(m) = g^m

Discrete log hardness $\Rightarrow$ privacy from eavesdropper?

But how will Bob figure out m?

- *He has to solve the same discrete log problem!*

Back to Diffie-Hellman Key Exchange

Alice

Bob

$(N, g), A = g^a$

$B = g^b$

$C = M.g^{ab}$

sample a

sample b

Maybe PK = (N, g, A), CT = (B, C)

ElGamal Public Key Encryption (1985)

Idea: Instead of sending (N, g), g^a just to Bob,
Alice publishes this as her public key PK

$$PK = (N, g, g^a)$$

- Keeps a as her secret key SK

To encrypt M: Bob does exactly as in DH KE. Bob samples random b, computes g^{ab}, and uses it to mask the message

$$Enc(M, PK) = (g^b, M.g^{ab})$$

To decrypt: Alice computes g^{ab} using g^b and a. Computes its inverse. Recovers M from $M.g^{ab}$

Security

<u>Adversary sees;</u>

$PK = (N, g, g^a)$
$C = (g^b, M.g^{ab})$

DDH Assumption: Given (N, g, g^a, g^b), can't distinguish g^{ab} from random

One can show: looks same as random

ElGamal Encryption is Randomized

To encrypt M: Bob samples random b, computes g^{ab}, and uses it to mask the message

$$Enc(M, PK) = (g^b, M.g^{ab})$$

Everytime b will be different. Hence, even if you encrypt the same M, you might get different ciphertexts!

Randomized Encryption?

Randomized encryption is a
feature rather than a bug

Deterministic encryption = bad security

ElGamal Public Key Encryption (1985)

It took 8+ years from the Diffie-Hellman
key exchange to the ElGamal encryption scheme

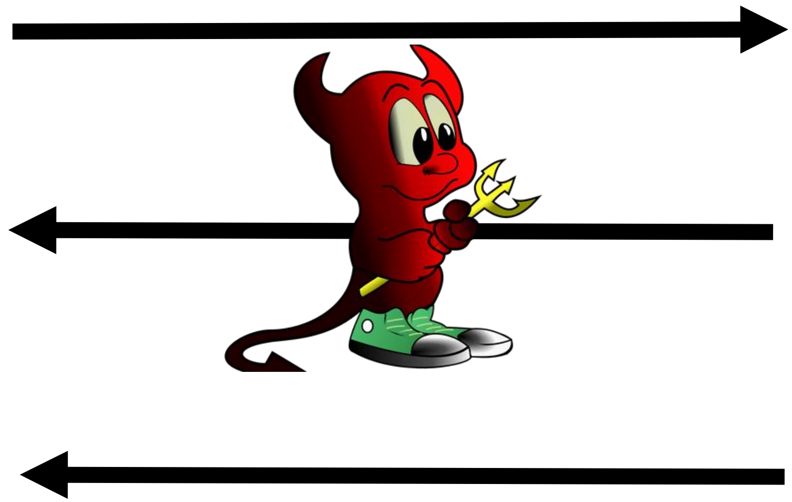
- In fact, this was not the first proposal for a PKE
- That honor belongs to the RSA scheme (1978)
- But ElGamal remains the simplest PKE

Putting Signatures and Encryption to Work: HTTPS/SSL protocol

How a new pair of parties could communicate?

You

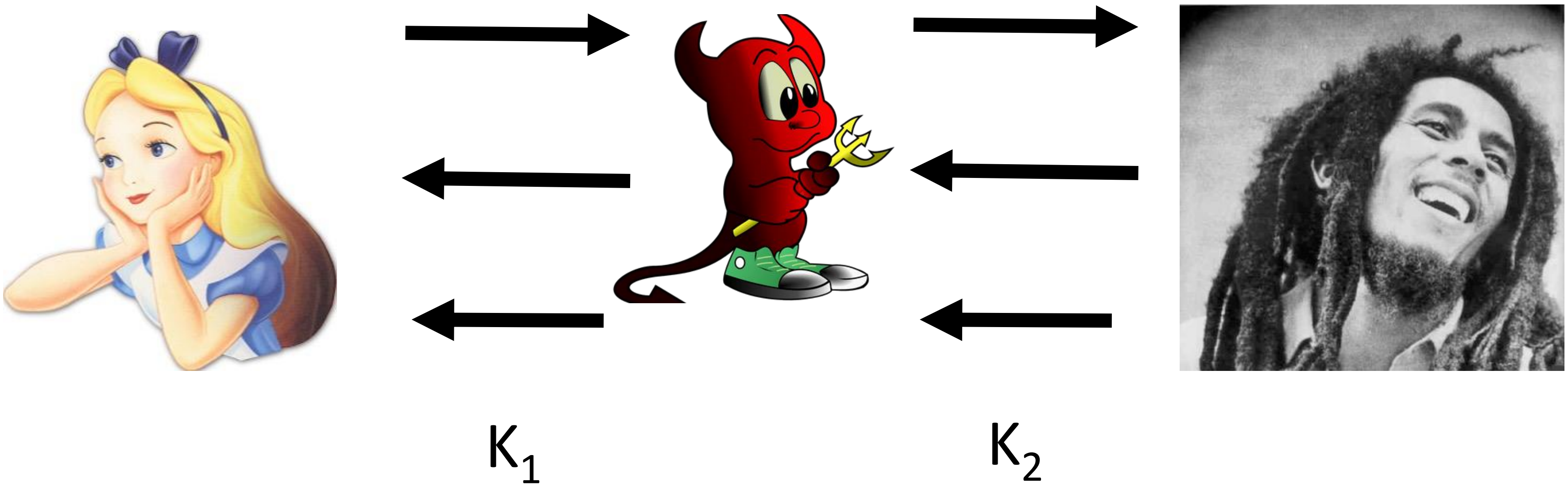
Google



DH KE? What if adv can modify messages?

Run key exchange with Alice and Bob separately!

How a new pair of parties could communicate?



Run key exchange with Alice and Bob separately!

Decrypt Alice's message using K_1, read, encrypt under K_2 and send to Bob!!

Use PKE?

You

PK

Google

Enc(PK, m)

Google sends you their PK, you encrypt?

Adv changes PK to PK_{adv}

Certificates and Certificate Authorities

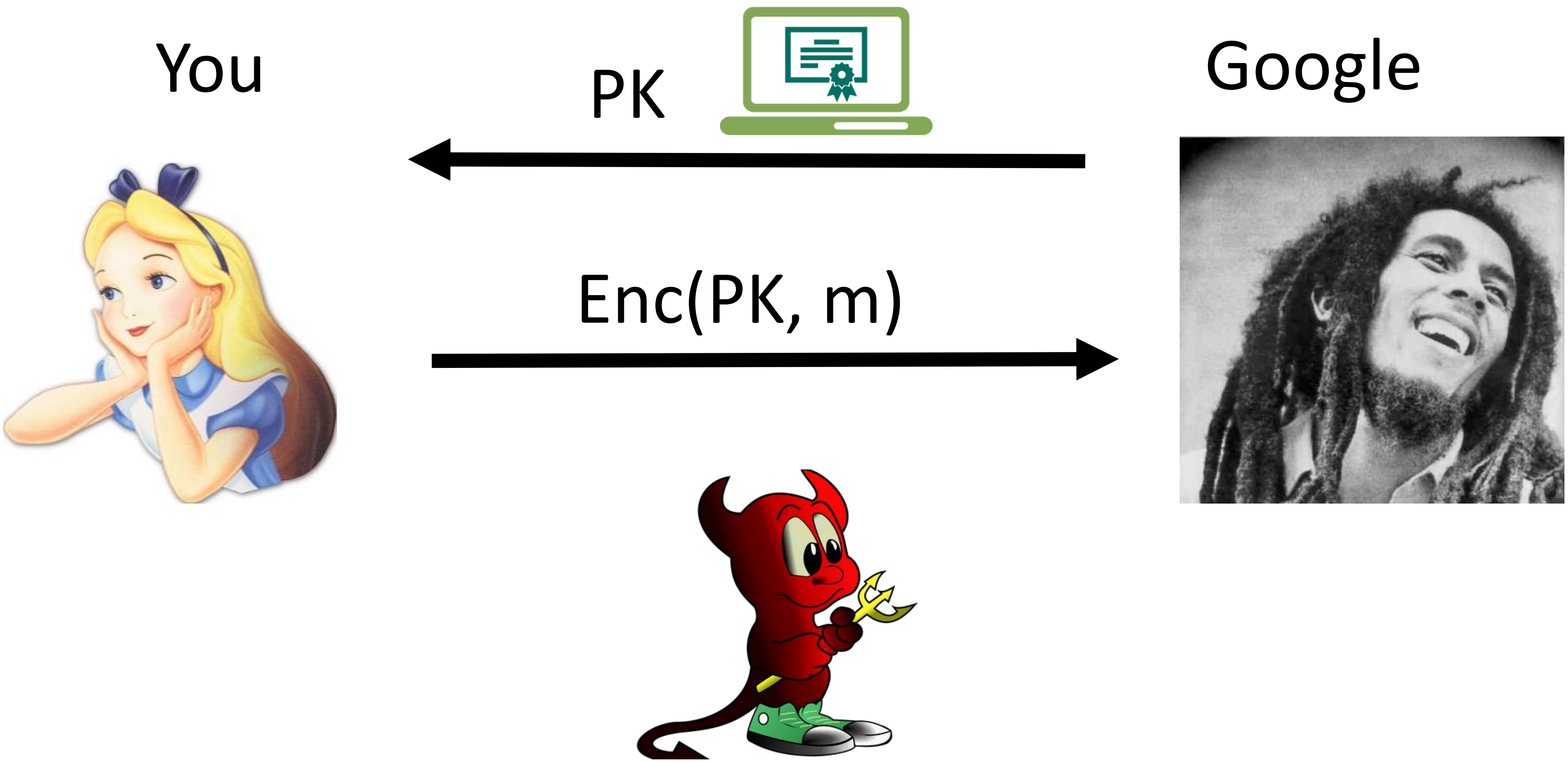
Authority

“PK is the public key of Google.com”

Digitally signed by authority

Google

HTTPS



Adversary can’t change certificate since its digitally signed

How do You Verify the Certificate?

$VK_{authority}$

$VK_{authority}$ is inside your browser/OS

Questions?