601.445/645
Practical Cryptographic Systems

Asymmetric Cryptography IV

Instructor: Matthew Green

Housekeeping

- A2
  - Due 23rd February, 11:59pm

News?

News?

‘Codefinger’ hackers encrypting Amazon cloud storage buckets

Cybercriminals have begun to encrypt data held in Amazon storage tools used by thousands of organizations around the globe.

Researchers with the cybersecurity firm Halcyon documented a recent trend of hackers going after Amazon Web Services’ cloud storage products known as S3 buckets and using the company’s own encryption tools to lock customers out of their data.

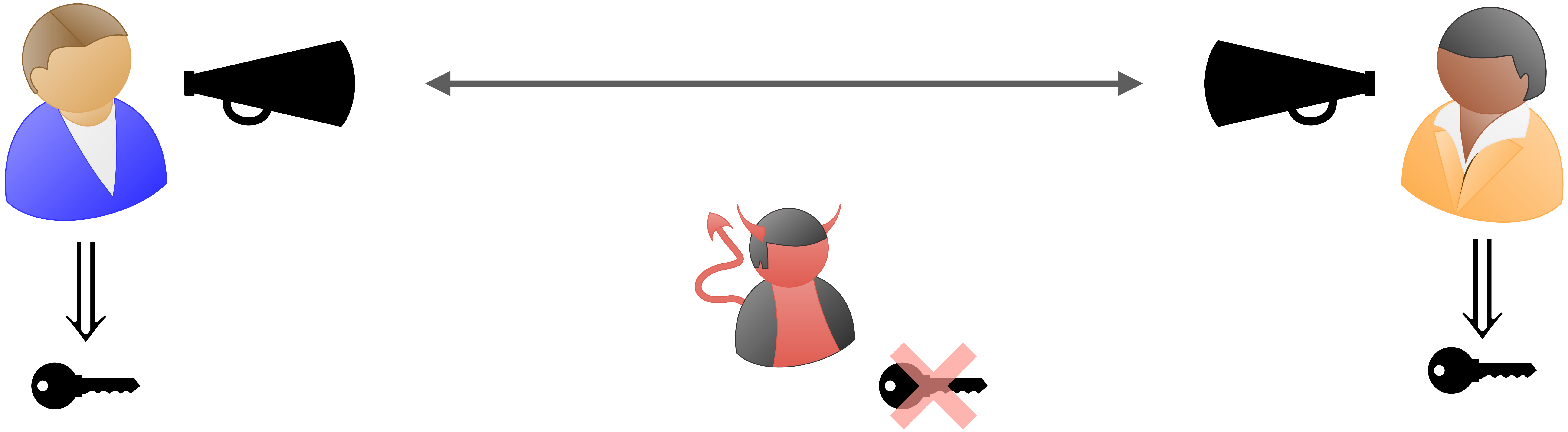
Halcyon has observed two such incidents since the beginning of December. The researchers dubbed the group behind the attack “Codefinger.”

“As they have only been observed in the two attacks noted in this report, Halcyon does not currently have any further intelligence on them, their origin, where they operate, or who they typically target,”a spokesperson told Recorded Future News. “Both victims were AWS native software developers.”

The attacks leverage Amazon Web Service’s server-side encryption with customer-provided keys (SSE-C) to encrypt customer data.

The hackers steal a customer’s AWS account credentials, obtain encryption keys and then lock customers out, demanding a ransom payment in exchange for the keys.

Key Exchange

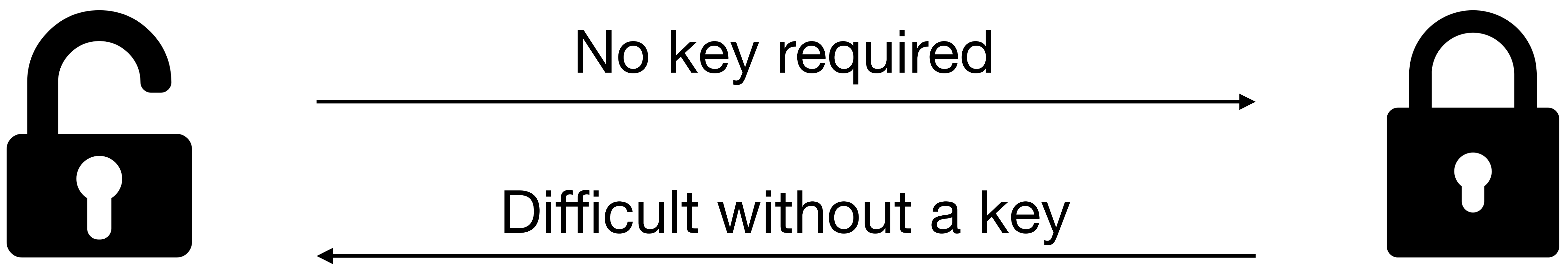


Diffie-Hellman Key Exchange

Agreeing on a common secret over an untrusted/public channel

Key Idea: Exploiting asymmetry

Often present in the real world!



Discrete Logarithm problem

- **<u>Discrete logarithm problem</u>**

Given: $x \in_R 0, \ldots, p-2$

$$\langle g \rangle = \mathbb{G} \qquad order(g) = p-1$$

$$h = g^x$$

Find: x

This problem is <u>hard</u> if for all p.p.t. adversaries, all attackers find x with "small" probability

See Boneh-Shoup, Appendix A

Discrete Logarithm problem

This means that "reversing" exponentiation is assumed to have super-polynomial running time.

How about the exponentiation itself?

- **<u>Discrete logarithm problem</u>**

Given: $x \in_R 0, \ldots, p-2$

$$\langle g \rangle = \mathbb{G} \qquad order(g) = p - 1$$

$$h = g^x$$

Find: x

This problem is hard if for all p.p.t. adversaries, all attackers find x with "small" probability

See Boneh-Shoup, Appendix A

Discrete Logarithm problem

This means that "reversing" exponentiation is assumed to have super-polynomial running time.

How about the exponentiation itself?

- **Discrete logarithm problem**

Given: $x \in_R 0, \ldots, p-2$

$\langle g \rangle = \mathbb{G}$ $\qquad order(g) = p$

$h = g^x$

Note that for this to hold, the size of *p* must be pretty large!

In practice, we typically assume *p* is at least 1024 bits. And 3072 bits is the minimum in modern protocols!

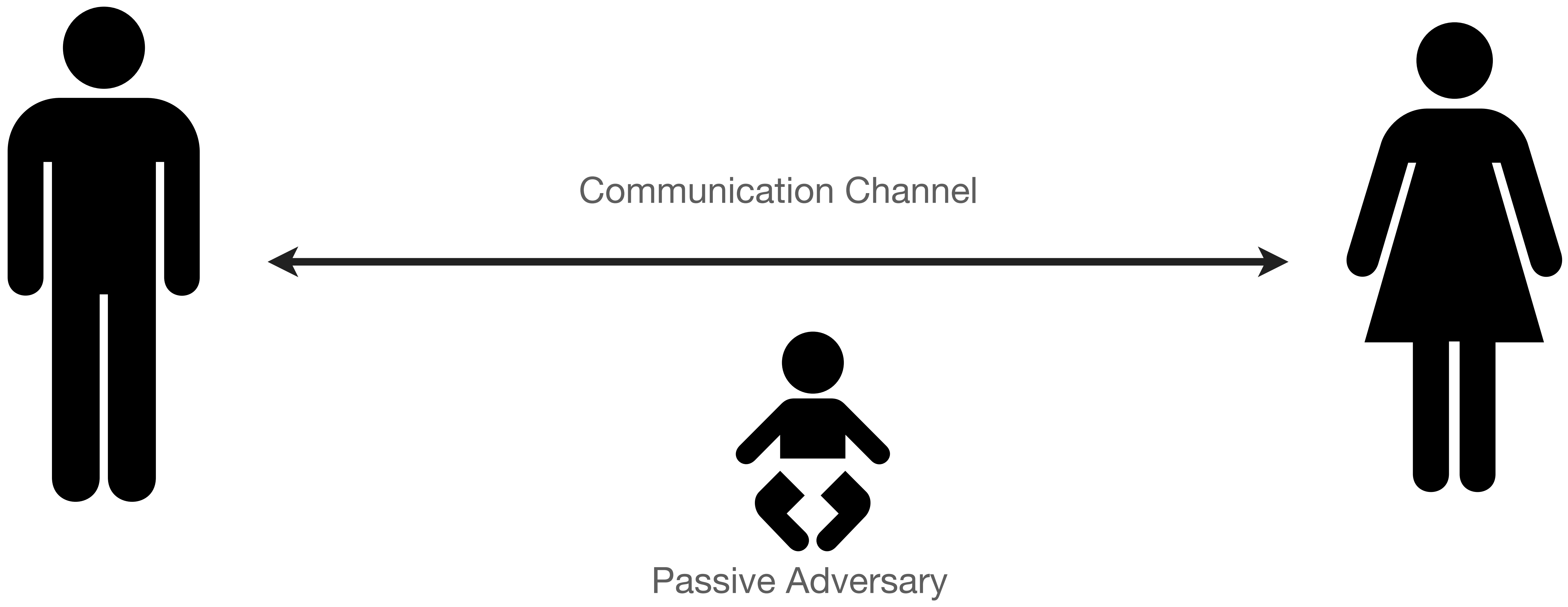
Find: x

This problem is hard if for all p.p.t. adversaries, all attackers find x with "small" probability

See Boneh-Shoup, Appendix A

Key Agreement

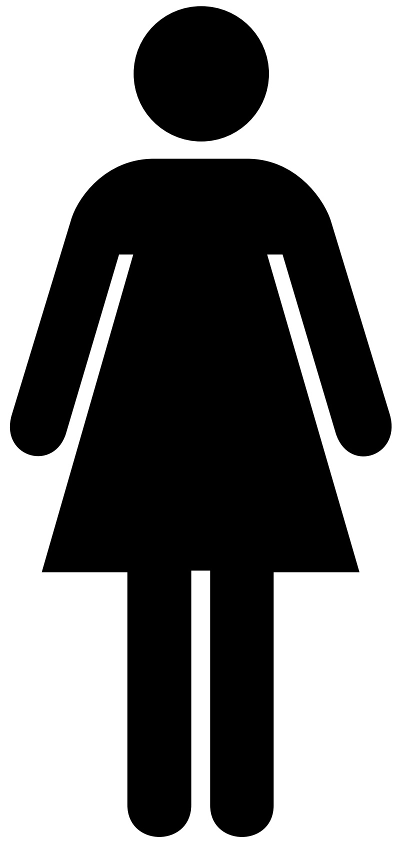
- Establish a shared key in the presence of a passive adversary

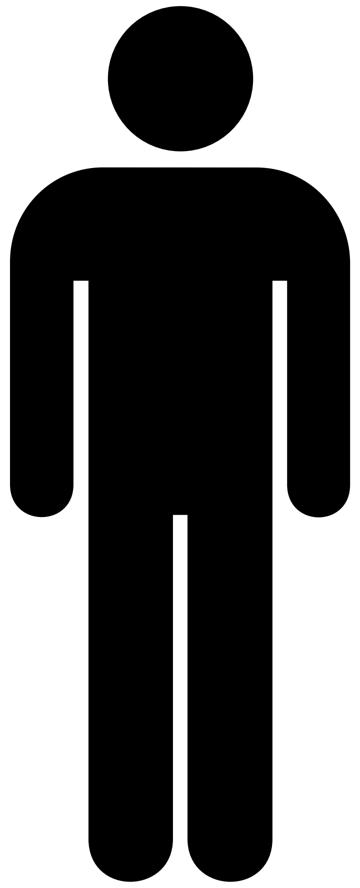


D-H Protocol

$$p, \langle g \rangle = \mathbb{Z}_p^*$$

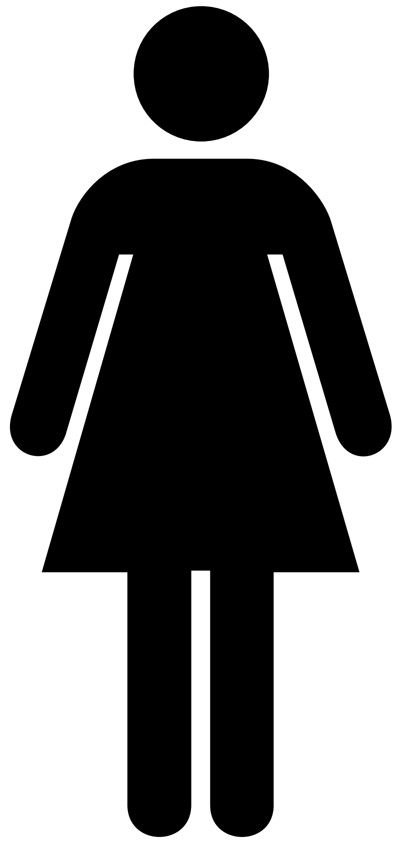
$$a \in \mathbb{Z}_{\phi(p)}$$



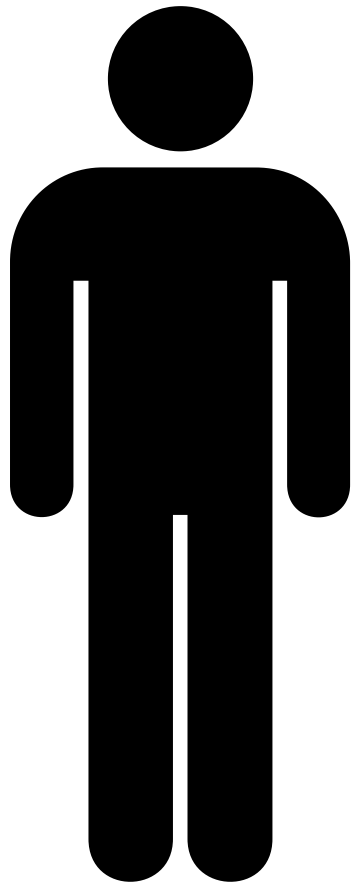


D-H Protocol

$$p, \langle g \rangle = \mathbb{Z}_p^*$$
$$a \in \mathbb{Z}_{\phi(p)}$$



$$\xrightarrow{p, g, g^a}$$



D-H Protocol

$p, \langle g \rangle = \mathbb{Z}_p^*$

$a \in \mathbb{Z}_{\phi(p)}$

$b \in \mathbb{Z}_{\phi(p)}$

$\xrightarrow{p, g, g^a}$

$\xleftarrow{g^b}$

D-H Protocol

$p, \langle g \rangle = \mathbb{Z}_p^*$

$a \in \mathbb{Z}_{\phi(p)}$

$b \in \mathbb{Z}_{\phi(p)}$

p, g, g^a

g^b

g^{ba}

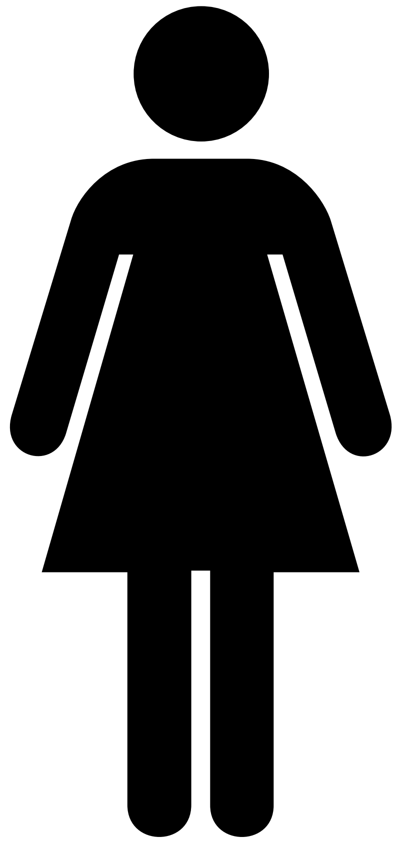
g^{ab}

D-H Protocol

$p, \langle g \rangle = \mathbb{Z}_p^*$

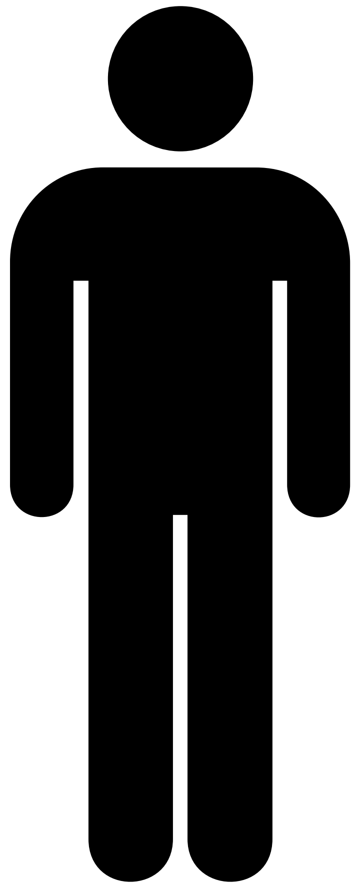
$a \in \mathbb{Z}_{\phi(p)}$

$b \in \mathbb{Z}_{\phi(p)}$



$$\xrightarrow{p,\, g,\, g^a}$$

$$\xleftarrow{g^b}$$



g^{ba}

g^{ab}

Usually we "hash" the shared secret value to make a secret encryption key, and then encrypt using a fast symmetric encryption scheme!

Hard problems (2)

- **<u>Diffie-Hellman problem</u>**

Given: $a, b \in_R 0, \ldots, p-2$

$$\langle g \rangle = \mathbb{G} \qquad order(g) = p-1$$

$$(g, g^a, g^b)$$

Find: g^{ab}

This problem is <u>hard</u> if for all p.p.t. adversaries, all attackers output a solution with “small” probability

See Boneh-Shoup, Appendix A

Hard problems (2)

- **<u>Diffie-Hellman problem</u>**

Notice this is just the Diffie-Hellman scheme re-written as a mathematical assumption!

Given: $a, b \in_R 0, \ldots, p-2$

$$\langle g \rangle = \mathbb{G} \qquad order(g) = p - 1$$

$$(g, g^a, g^b)$$

Find: g^{ab}

This problem is <u>hard</u> if for all p.p.t. adversaries, all attackers output a solution with “small” probability.

See Boneh-Shoup, Appendix A

Hard problems (2)

- **<u>Diffie-Hellman problem</u>**

Given: $a, b \in_R 0, \ldots, p-2$

$$\langle g \rangle = \mathbb{G} \qquad order(g) = p - 1$$

$$(g, g^a, g^b)$$

Find: g^{ab}

This problem is <u>hard</u> if for all p.p.t. adversaries, all attackers output a solution with “small” probability.

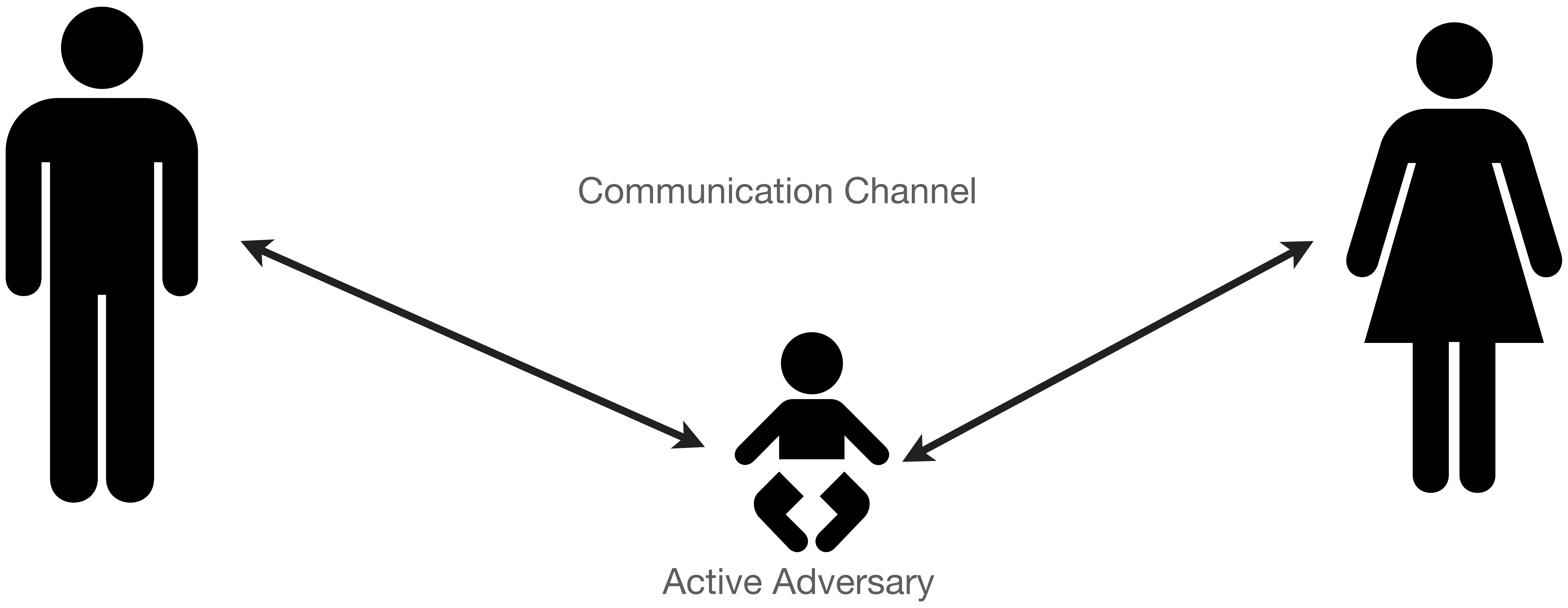
Notice this is just the Diffie-Hellman scheme re-written as a mathematical assumption!

Note that for this to hold, the size of *p* must be pretty large!

In practice, we typically assume *p* is at least 1024 bits. And 3072 bits is the minimum in modern protocols!

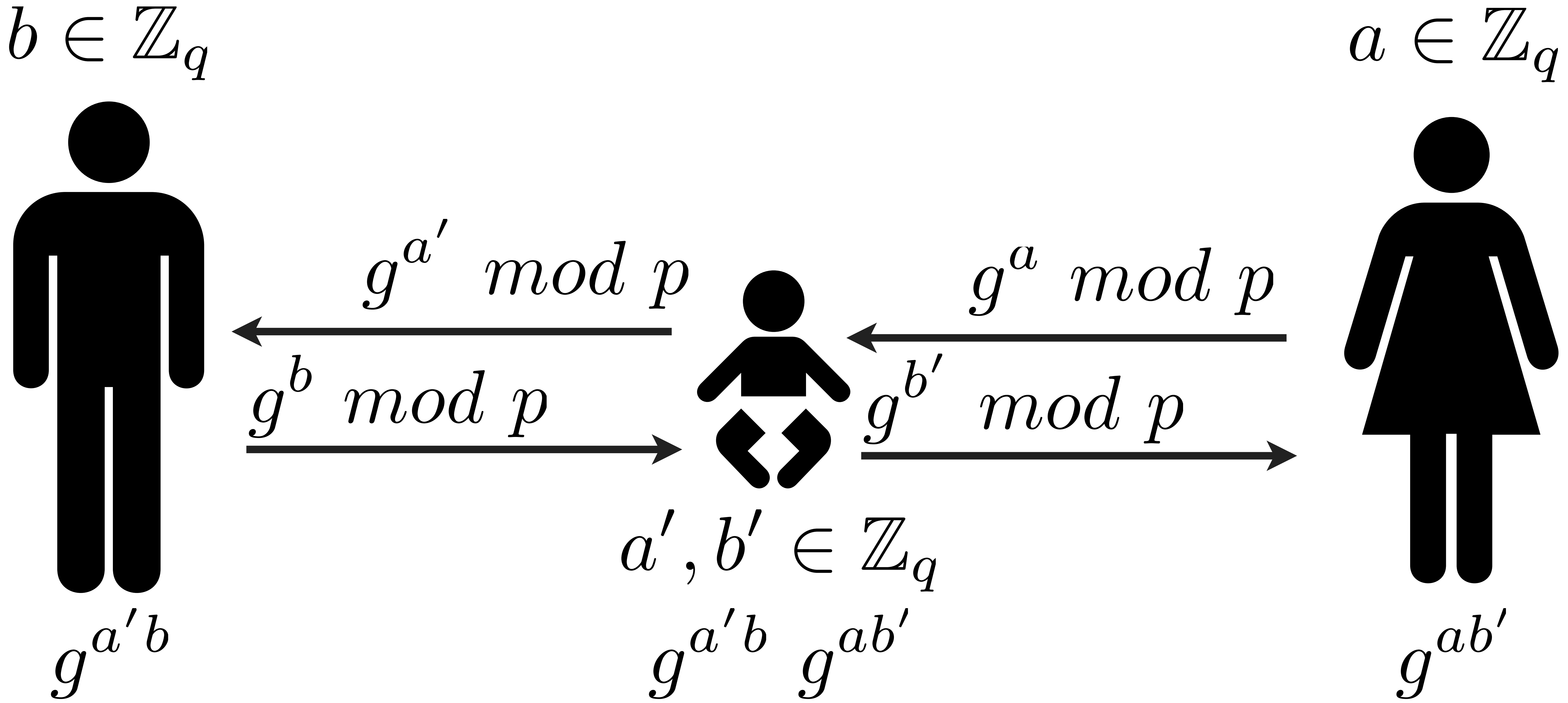
See Boneh-Shoup, Appendix A

What if we have an active adversary?



Man in the Middle

- Assume an active adversary:



Man in the Middle

- Caused by lack of <u>authentication</u>
  - D-H lets us establish a shared key with anyone...
    but that's the problem…
  - We don't know if the person we're talking to is the right person
- Solution?

Preventing MITM

- Verify key via separate channel
- Password-based authentication
- Authentication via PKI

Digital Signatures

- Similar to MACs, with public keys
  - Secret key used to sign data
  - Public key can verify signature
  - Advantages over MACs?

Digital Signatures

- Three algorithms:

  Keygen() -> (vk, sk)

  Sign(sk, message) -> sig

  Verify(pk, message, sig) -> True/False

Signature: definitions

- Existential Unforgeability under Chosen Message Attack
  - No efficient adversary can “forge” a signature on any new message (that it hasn’t received a signature for)
  - Even if it has access to an “oracle” that signs any message it wants
  - “Strong unforgeability” <- can’t even make a new signature for an existing message

Certificates


```
-----BEGIN CERTIFICATE-----
MIID9TCCA16gAwIBAgIJAP5UpXOKgZ+sMA0GCSqGSIb3DQEBBQUAMIGuMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCQ0ExFjAUBgNVBAcTDVNhbiBGcmFuY2lzY28xJDAi
BgNVBAoTG1Rlc3QgQ2VydGlmaWNhdGUgSW5kdXN0cmllczEQMA4GA1UECxMHVGVz
dGluZzEZMBcGA1UEAxMQQWxidXMgRHVtYmxlZG9yZTEnMCUGCSqGSIb3DQEJARYY
ZG8tbm90LXJlcGx5QGRyb3Bib3guY29tMB4XDTEzMTIwNTIxMTQyOVoXDTE0MTIw
NTIxMTQyOVowga4xCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDQTEWMBQGA1UEBxMN
U2FuIEZyYW5jaXNjbzEkMCIGA1UEChMbVGVzdCBDZXJ0aWZpY2F0ZSBJbmRlc3Ry
aWVzMRAwDgYDVQQLEwdUZXN0aW5nMRkwFwYDVQQDExBBbGJ1cyBEdW1ibGVkb3Jl
MScwJQYJKoZIhvcNAQkBFhhkby1ub3QtcmVwbHlAZHJvcGJveC5jb20wgZ8wDQYJ
KoZIhvcNAQEBBQADgY0AMIGJAoGBALUgT5viXElXI4BdxyhoR8Y4VUdyAsqv0C/u
cDU9GhMkc0S2jhjNMtThg3As9mbTo7x2ITwXpAgTBUvXzNmaV6HXhK8MASMBwAGo
1K5P3/JidTmWaIPo+eOfjr9/HtOhSiO17HQQBoV9flt6kYGoD6nXqgt1Y8B1lZ3a
ZtRKlc6VAgMBAAGjggEXMIIBEzAdBgNVHQ4EFgQUZrz7ayaUDn+t7ekkc64HqnCR
L1wwgeMGA1UdIwSB2zCB2IAUZrz7ayaUDn+t7ekkc64HqnCRL1yhgbSkgbEwga4x
CzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDQTEWMBQGA1UEBxMNU2FuIEZyYW5jaXNj
bzEkMCIGA1UEChMbVGVzdCBDZXJ0aWZpY2F0ZSBJbmRlc3RyaWVzMRAwDgYDVQQL
EwdUZXN0aW5nMRkwFwYDVQQDExBBbGJ1cyBEdW1ibGVkb3JlMScwJQYJKoZIhvcN
AQkBFhhkby1ub3QtcmVwbHlAZHJvcGJveC5jb22CCQD+VKVzioGfrDAMBgNVHRME
BTADAQH/MA0GCSqGSIb3DQEBBQUAA4GBAGRrWit1A8EbETqaM1Aue938+K1IBM26
bK904jrtSypH/t/uoO5hpR+AzXlmRLpdXw2CgqpQzZIxB0zM7YZR10x05HL4yRRx
8V7v/8keeiRqA9o3XUw2FyvYkn+HZYdReGp8pECcmj5GD4FgCyK6GXo5/xzoMo7o
01IVrbOFCXM2
-----END CERTIFICATE-----
```

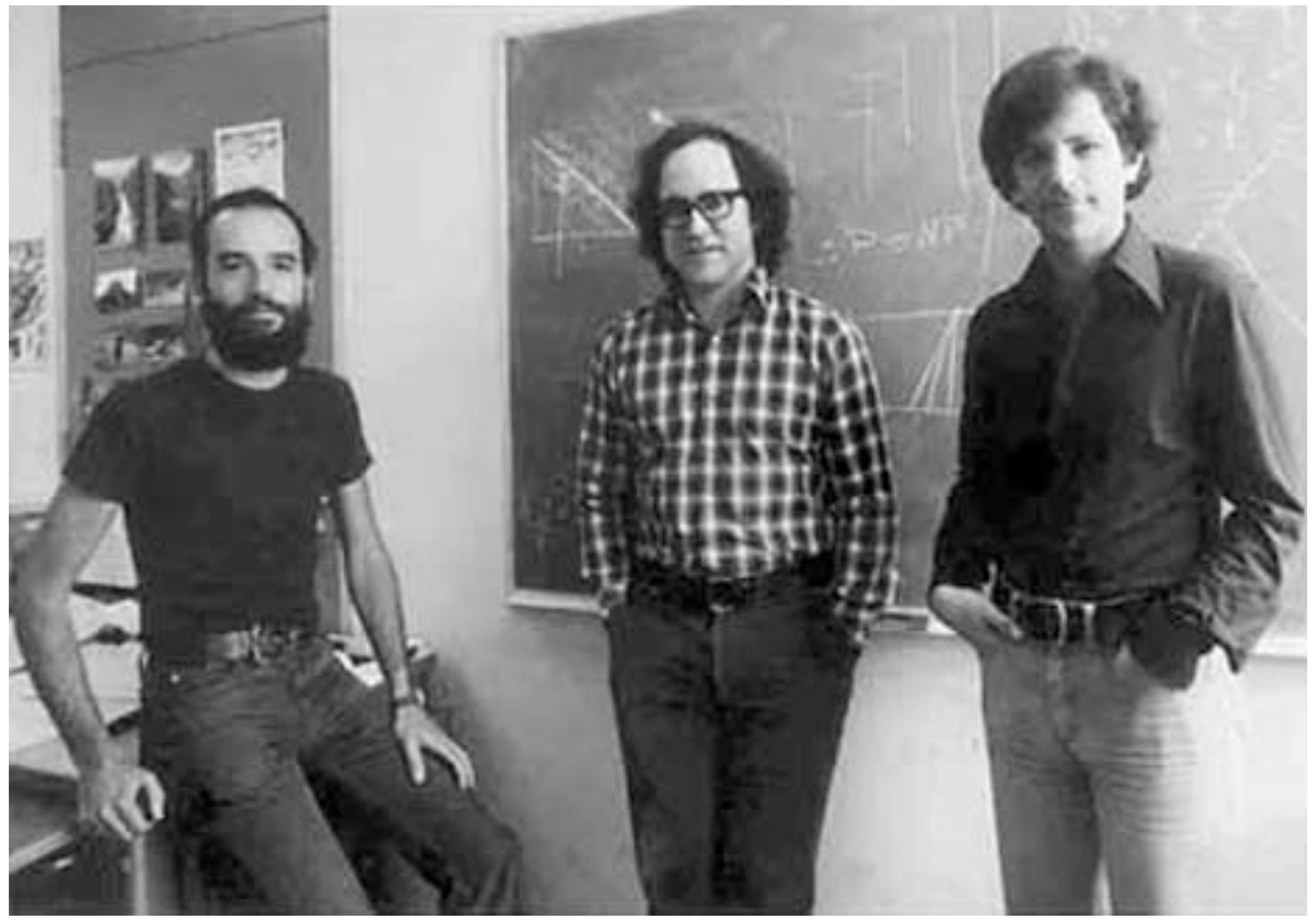
Certificates

Public Key Info

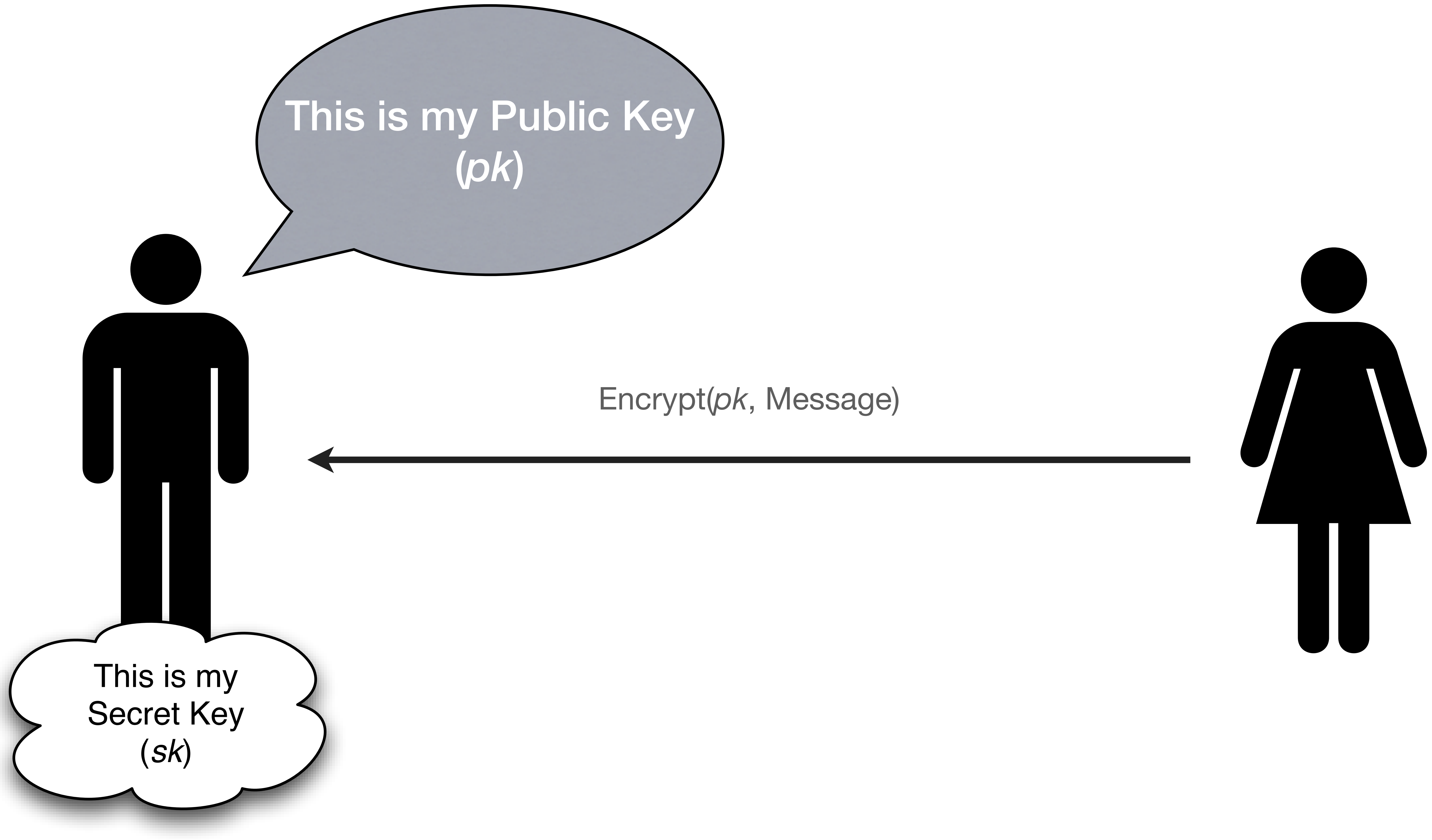
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	256 bytes : AD 0F EF C1 97 5A 9B D8 1E B0 44 8D C6 C9 A0 28 C3 0E 68 1B 94 91 2E 77 EC AC AE BE 6C 78 04 5B A4 78 04 CE FB 07 4B 5D 34 F3 57 E5 0F FB 6B A4 2A A5 53 D3 D5 7F 3A 3C 54 4C EB 73 7B 5E A1 0A D9 7E 5F A9 5A C0 71 71 43 9D 6F BD 4C CC CC 43 8C CF 77 4B 9D 1A 75 CB 1F BD F7 3B D3 66 C6 CE 7C B0 5A FC D4 14 24 3A 2A C5 A8 61 6D 04 4D A6 36 2D B0 FC C4 B0 BF FC 41 27 71 E4 C3 90 AD 37 07 67 BE 5A 1A 81 9D AB 8A 71 92 A3 85 1D 99 E7 20 19 CF C4 FD AD 9F 6E 98 9F 5B CE 17 A1 FE 7B 4A 4F C9 F2 AD 21 C8 F7 1B 5D 10 79 59 85 DF 7E B8 A8 FE 3A D7 2F E2 02 DF D8 67 67 F4 63 9F FA B3 E7 47 63 48 3A C1 98 73 3D 9A 8D 8D DA AC C8 DF 50 32 BC A1 21 A6 10 56 AE E6 C6 10 2A 4E 54 41 5D 38 C1 37 77 78 1E 43 F8 70 2A 4B 4D EA B7 F9 51 CC 1C 17 4F 2A 1B 67 1C 2E E0 E0 2D 7C 59
Exponent	65537
Key Size	2,048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	512 bytes : 36 07 E7 3B B7 45 97 CA 4D 6C B0 2A 3F 3F 38 43 12 3D 1C 4C 8E F6 87 18 5C 66 54 C5 E2 5B 4B ED ED DC 4C 23 EC 93 21 A1 19 28 DD 78 6D A6 0D E7 F4 F5 64 2E 1B 49 22 B4 EE FE E7 D3 0B 34 85 6A 12 14 09 33 4F 4E 52 FD 6B B0 04 9A EF 62 3C E3 78 6C 08 7A 87 25 63 61 28 B2 2C 22 10 5E 51 0F 03 7B 53 41 48 74 47 7D 3C 06 C3 E6 56 4D 96 9C 09 62 B2 76 00 9F 1A 3C C8 08 67 05 A1 C1 55 48 C2 37 EA 32 69 6A 12 E2 53 26 DB AC AB 79 94 88 8B 5B 5A 72 76 04 76 0D 53 CC 3D A9 38 95 E6 C1 BE E0 A4 C8 7E F6 AC 7E FF 34 ED 3B 5D 38 46 67 1C C5 79 D4 A8 81 8E 9C D0 CA F7 75 64 4F DC F8 4A 38 7C 88 18 DC D1 9B 50 F1 DB E8 61 D4 7D AE D8 9E 6E 86 E9 73 4A D4 2A F1 C7 CA 69 19 89 56 B5 FC BE 8D 90 F4 5A 21 89 A4 9A B7 3B F5 BA 24 34 A0 FD 5E 59 80 7A 45 93 3B 56 89 62 E3 4E E3 7E EB 13 2B 28 24 B9 86 EC DA 93 49 A1 0F 14 EF 54 93 BE 1E F4 55 CF 17 20 C5 01 C5 84 62 D5 64 38 1D 1C 59 08 D1 31 F8 AE 05 A4 1B BA 0A 67 51 9E A8 15 F2 E8 CF 8E 9E D8 88 52 21 89 CC 4F 98 13 0A 41 40 71 69 79 B0 A5 6A BE 77 AB 5E A1 D4 89 66 6C 02 C2 D1 43 0D A2 CA D7 7A 71 01 8B F7 98 21 74 89 E8 8B 27 38 28 CD 3E EA A7 78 AD 2A 3A 63 DB 3A D0 05 6B 4F C9 20 4E 01 38 DF 05 75 49 F7 9F 2E DC 19 31 A9 96 D7 2F 2D 4E 84 7C FA 7E F6 67 5A A1 E7 5C A1 72 3B 22 DC A5 FA F2 E7 DC D6 A8 6D A0 4D FD 78 C5 5C DC 34 D9 86 76 5B 1C 0D BB B1 E5 DB 64 2A 55 7F 20 4D 5D 4D 44 01 1D 79 A3 2D EC F5 6B CD BE 7B 52 67 1D FF 05 42 FB 42 7A A1 BC 4C 23 DF AF 16 B9 76 C9 69 86 02 34 F2 A9 CB B8 15 39 BA A5 F1 E6 72 7C 1D 5E 0C 48 D7 99 1F 50 98 2B 75 2D 67 58 79 A1 1A 05 5A

Public Key Encryption

- What if our recipient is offline?
  - Key agreement protocols are interactive
  - e.g., want to send an email



Public Key Encryption



Public key encryption from D-H?

- Can we build public-key encryption from Diffie-Hellman?
- Idea: we will re-use the first move of the D-H protocol as a “public key”
  - Does this work?

RSA Cryptosystem

Choose large primes: p, q

$$N = p \cdot q$$

$$\phi(N) = (p-1)(q-1)$$

Choose:

$$e \;:\; gcd\,(e, \phi(N)) = 1$$

$$d \;:\; ed \; mod \; \phi(N) = 1$$

Output:

$$pk = (e, N)$$

$$sk = d$$

Encryption

$$c = m^e \; mod \; N$$

Decryption

$$m = c^d \; mod \; N$$

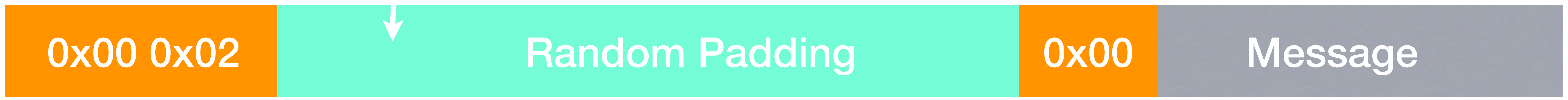
“Textbook RSA”

- In practice, we don’t use Textbook RSA
  - Fully deterministic (not semantically secure)
  - Malleable
  - Might be partially invertible

- Coppersmith’s attack: recover part of plaintext (when m and e are small)

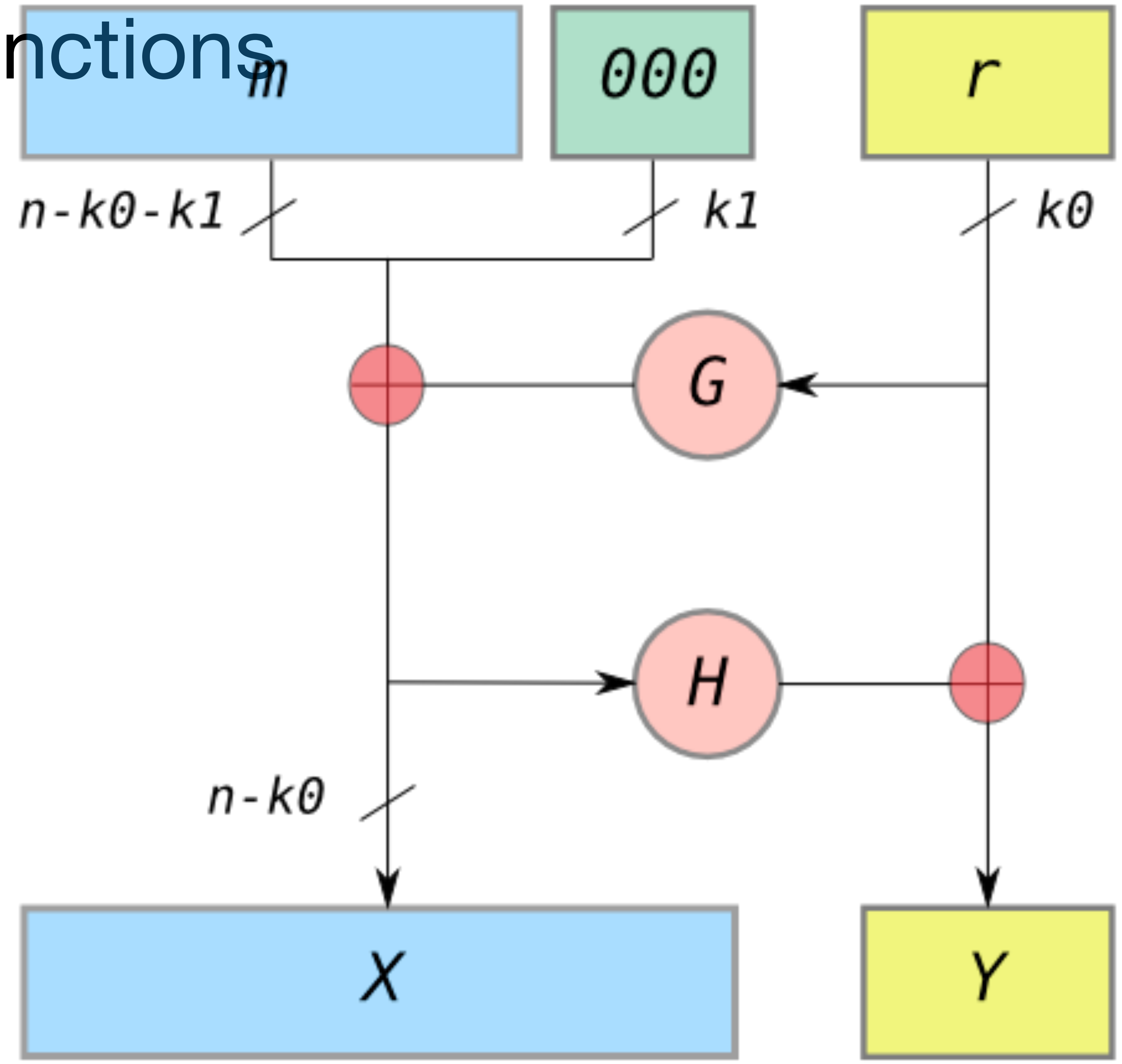
RSA Padding

- Early solution (RSA PKCS #1 v1.5):
  - Add “padding” to the message before encryption
  - Includes randomness
  - Defined structure to mitigate malleability
  - <u>PKCS #1 v1.5 badly broken</u> (Bleichenbacher)



RSA Padding

- Better solution (RSA-OAEP):
  - G and H are hash functions



Efficiency

	Cycles/Byte
AES (128 bit key)	18
DES (56 bit key)	51
RSA (1024 bit key) <u>Encryption</u>	1,016
RSA (1024 bit key) <u>Decryption</u>	21,719

Hybrid Encryption

- Mixed Approach
  - Use PK encryption to encrypt a symmetric key
  - Use (fast) symmetric encryption on data

Key Strength

Level	Protection	Symmetric	Asymmetric	Discrete Logarithm		Elliptic Curve	Hash
				Key	Group		
1	Attacks in "real-time" by individuals *Only acceptable for authentication tag size*	32	-	-	-	-	-
2	Very short-term protection against small organizations *Should not be used for confidentiality in new systems*	64	816	128	816	128	128
3	Short-term protection against medium organizations, medium-term protection against small organizations	72	1008	144	1008	144	144
4	Very short-term protection against agencies, long-term protection against small organizations *Smallest general-purpose level,* *Use of 2-key 3DES restricted to 2^{40} plaintext/ciphertexts,* *protection from 2009 to 2011*	80	1248	160	1248	160	160
5	Legacy standard level *Use of 2-key 3DES restricted to 10^{6} plaintext/ciphertexts,* *protection from 2009 to 2018*	96	1776	192	1776	192	192
6	Medium-term protection *Use of 3-key 3DES,* *protection from 2009 to 2028*	112	2432	224	2432	224	224
7	Long-term protection *Generic application-independent recommendation,* *protection from 2009 to 2038*	128	3248	256	3248	256	256
8	"Foreseeable future" *Good protection against quantum computers*	256	15424	512	15424	512	512