601.445/645 Practical Cryptographic Systems

Symmetric Cryptography/ Asymmetric Cryptography

Instructor: Matthew Green

Housekeeping

- Written assignment 1 out, 2pm
  - Due in 7 days (9/19, end of day) via BB
  - Available on Blackboard (for 445 section)
  - Please get it from Piazza (for 645 section)
    I promise JHU/BB is working on this!
  - A1 due at expected time

News?

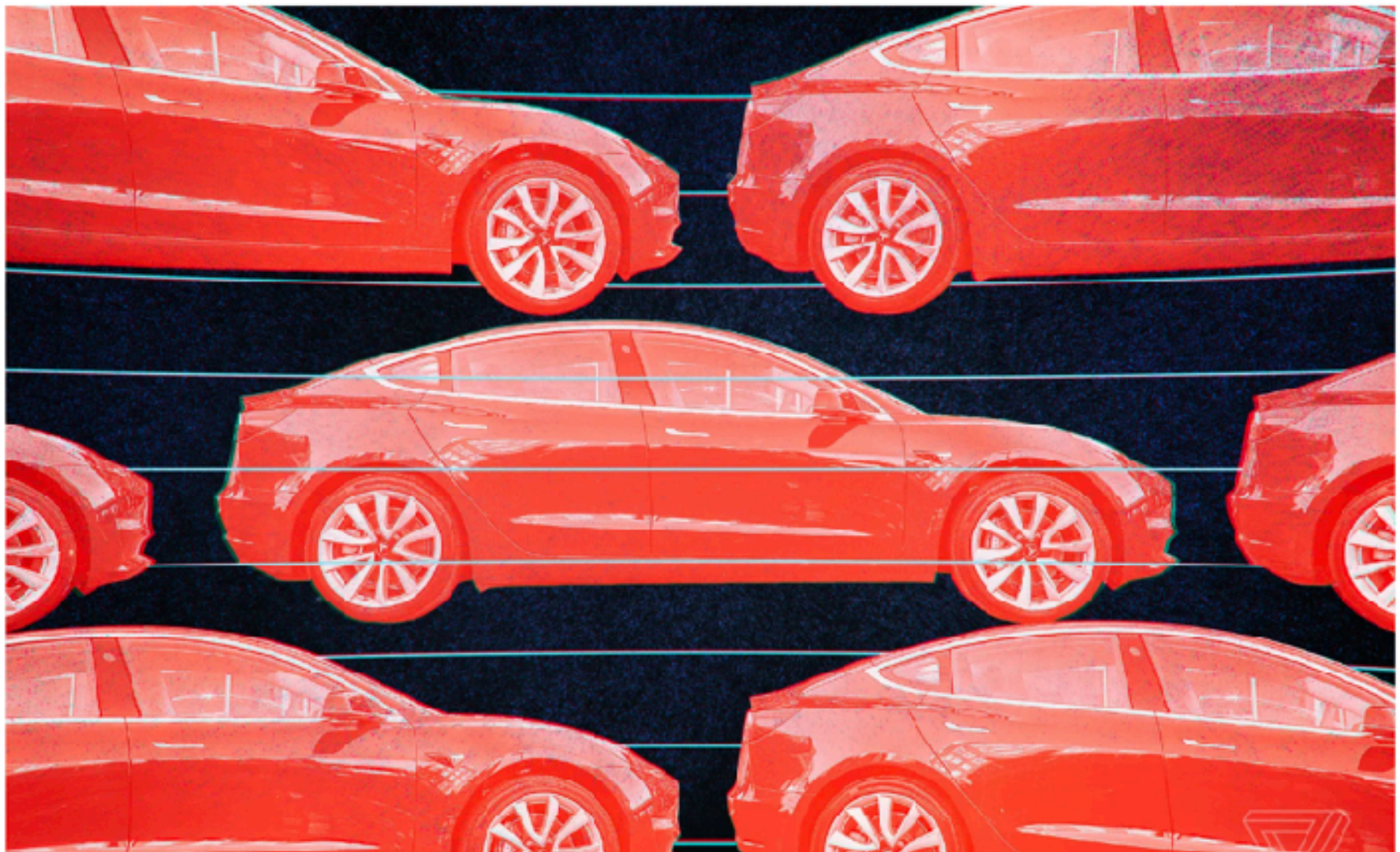
TECH \ TRANSPORTATION \ CARS

Tesla's keyless entry vulnerable to spoofing attack, researchers find

Now is a good time to add a PIN code to your Tesla

By Russell Brandom | @russellbrandom | Sep 10, 2018, 5:05pm EDT

SHARE



MOST READ

Apple leaks iPhone XS, XS Max, and XR names on its own website

Ne

Matthew Green @matthew_d_green · Sep 10

Finding a 40-bit cipher at the heart of any security system in 2018 is like opening up the Falcon Heavy and finding a bunch of Estes model rocket engines glued together.

5 83 284

Matthew Green @matthew_d_green · Sep 10

I can sympathize. In 2005 I was on a team that REed a similar system, and this also devolved to brute-forcing a 40-bit key. To see this happening in 2018 and in *Teslas*, though!

6 18 84

Cryp·tomer
@TomerAshur

Follow

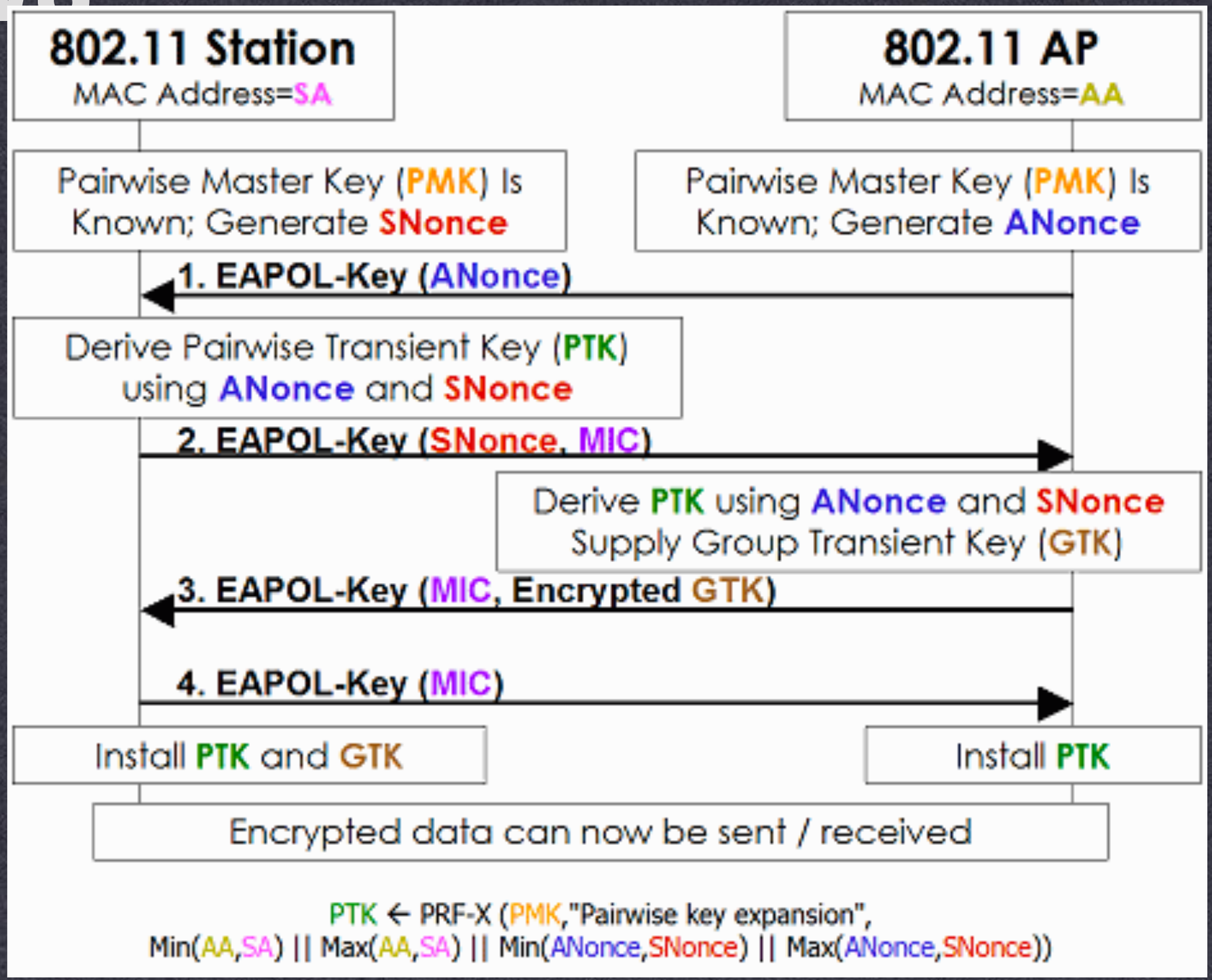
Replying to @matthew_d_green

Matt, it's the same system. We found that Tesla were using the same DST40 you reversed engineered in 2005.

12:45 PM - 10 Sep 2018

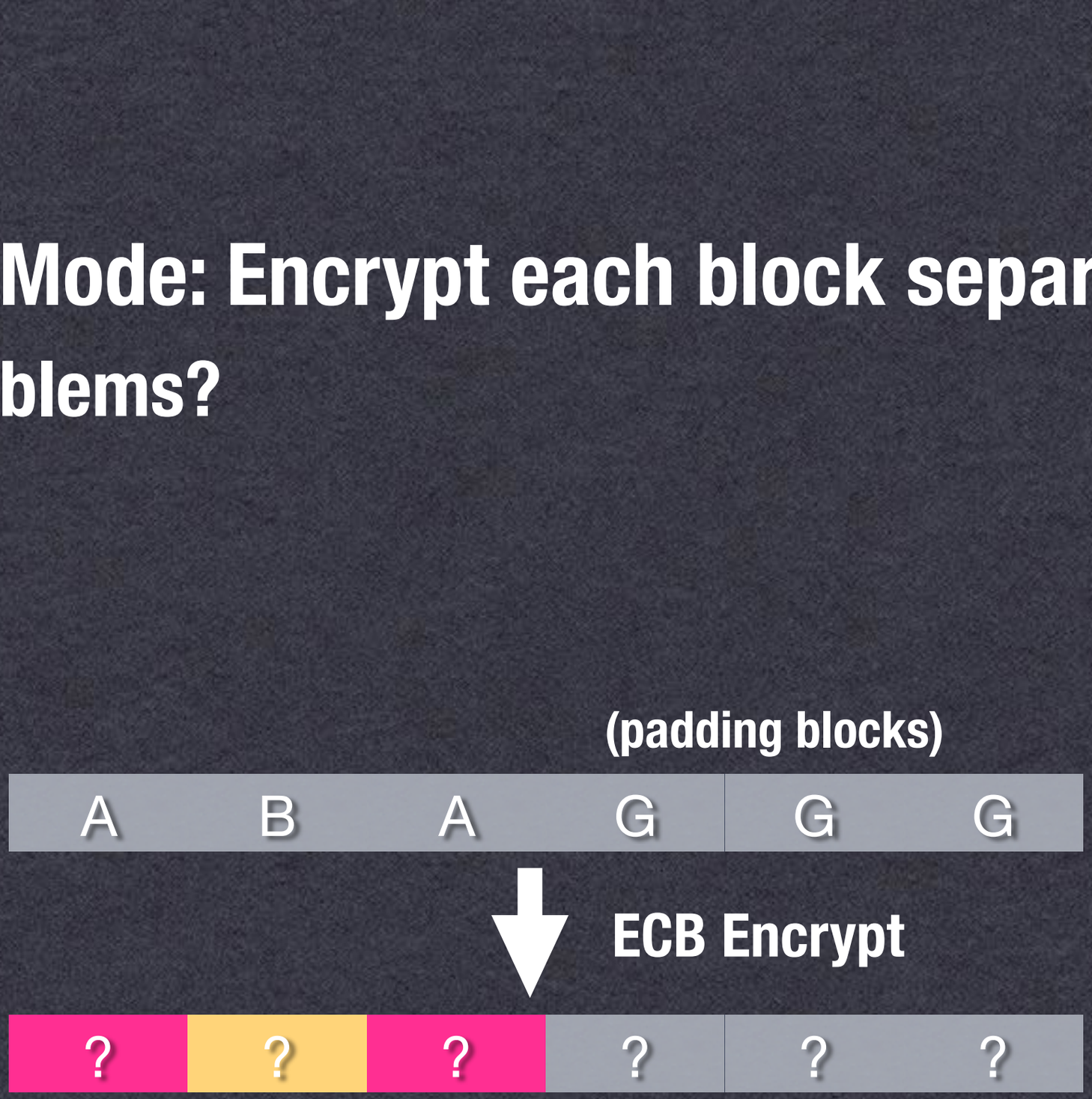
37 Retweets **176** Likes

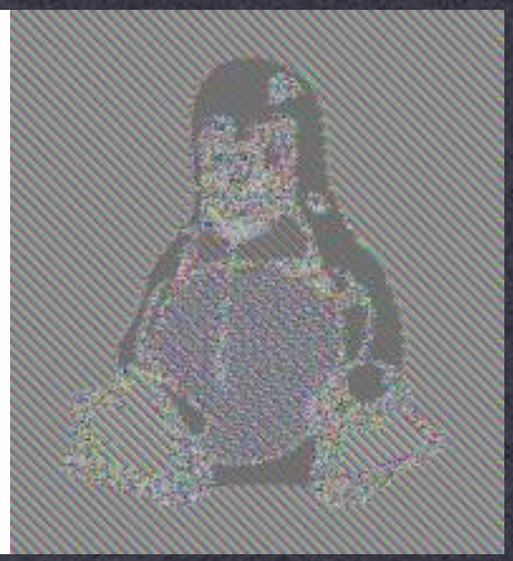
News


802.11 Station
MAC Address=SA
802.11 AP
MAC Address=AA
Pairwise Master Key (PMK) Is Known; Generate SNonce
Pairwise Master Key (PMK) Is Known; Generate ANonce
1. EAPOL-Key (ANonce)
Derive Pairwise Transient Key (PTK) using ANonce and SNonce
2. EAPOL-Key (SNonce, MIC)
Derive PTK using ANonce and SNonce
Supply Group Transient Key (GTK)
3. EAPOL-Key (MIC, Encrypted GTK)
4. EAPOL-Key (MIC)
Install PTK and GTK
Install PTK
Encrypted data can now be sent / received
PTK ← PRF-X (PMK,"Pairwise key expansion",
Min(AA,SA) || Max(AA,SA) || Min(ANonce,SNonce) || Max(ANonce,SNonce))

ECB

- ECB Mode: Encrypt each block separately
  - Problems?





Security of Encryption

- **Semantic Security**
  - Due to Goldwasser & Micali (1980s)
  - Informally: An encryption scheme is secure if adversary who sees ciphertext "learns as much"
    as adversary who doesn't see ciphertext.

-Even if adversary can request chosen plaintexts

  - How do we state this formally?

Review

- **Semantic Security (IND-CPA)**

$b \xleftarrow{\$} \{0,1\}$

k

Adversary

M_0, M_1

$E(M_b)$

b?

Review

- **Semantic Security (IND-CPA)**

$b \xleftarrow{\$} \{0, 1\}$

k

Adversary

$M_1, ..., M_n$

$E(k, M_1), ..., E(k, M_n)$

M_0, M_1

$E(M_b)$

$M_1, ..., M_n$

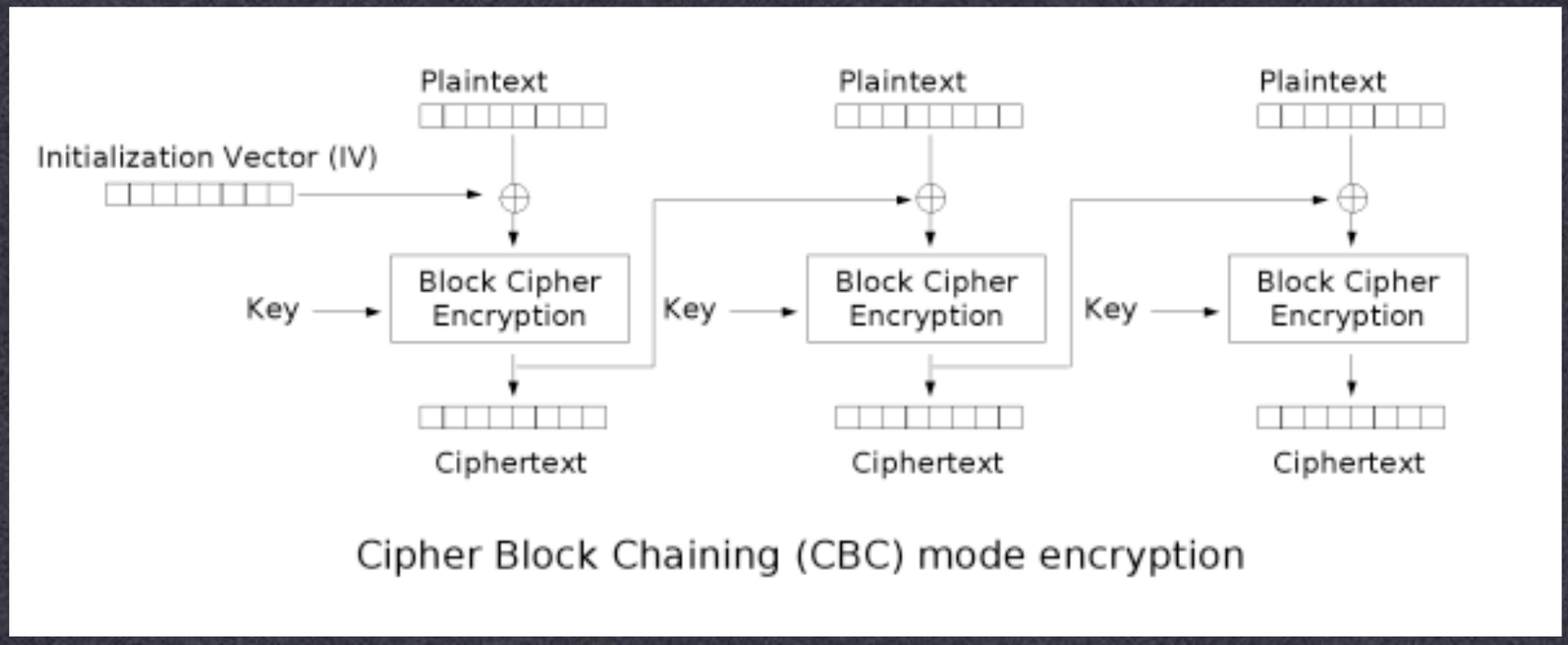
$E(k, M_1), ..., E(k, M_n)$

b?

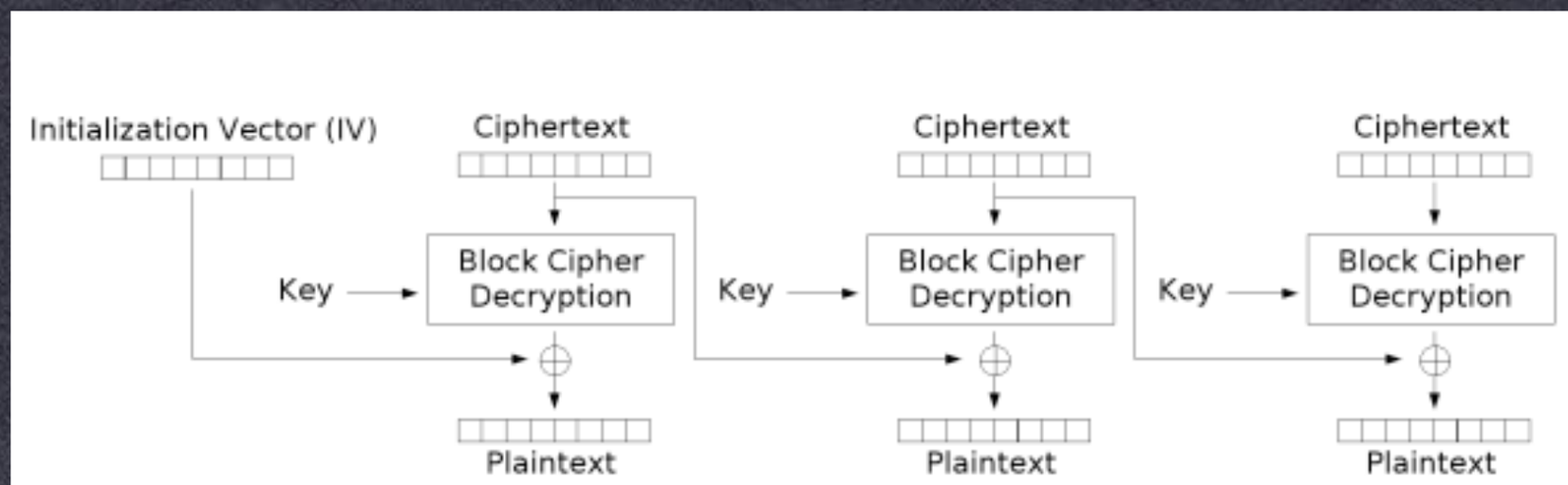
Using Block Ciphers

- ECB is not semantically secure, hence we use a "mode of operation"
  - e.g., CBC, CTR, CFB, OFB (and others)
- These provide:
  - Security for multi-block messages
  - Randomization (through an Initialization Vector)

CBC Mode



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Security of CBC

- Is CBC a secure encryption scheme?
  - Yes, assuming a secure block cipher
  - Correct (random) IV generation
  - Can <u>prove</u> this under assumption that block cipher = Pseudo-Random Permutation (PRP)

-Bellare, Desai, Jokipii & Rogaway (2000)

- Easy to use wrong...
  - Most important: use a unique & random IV!

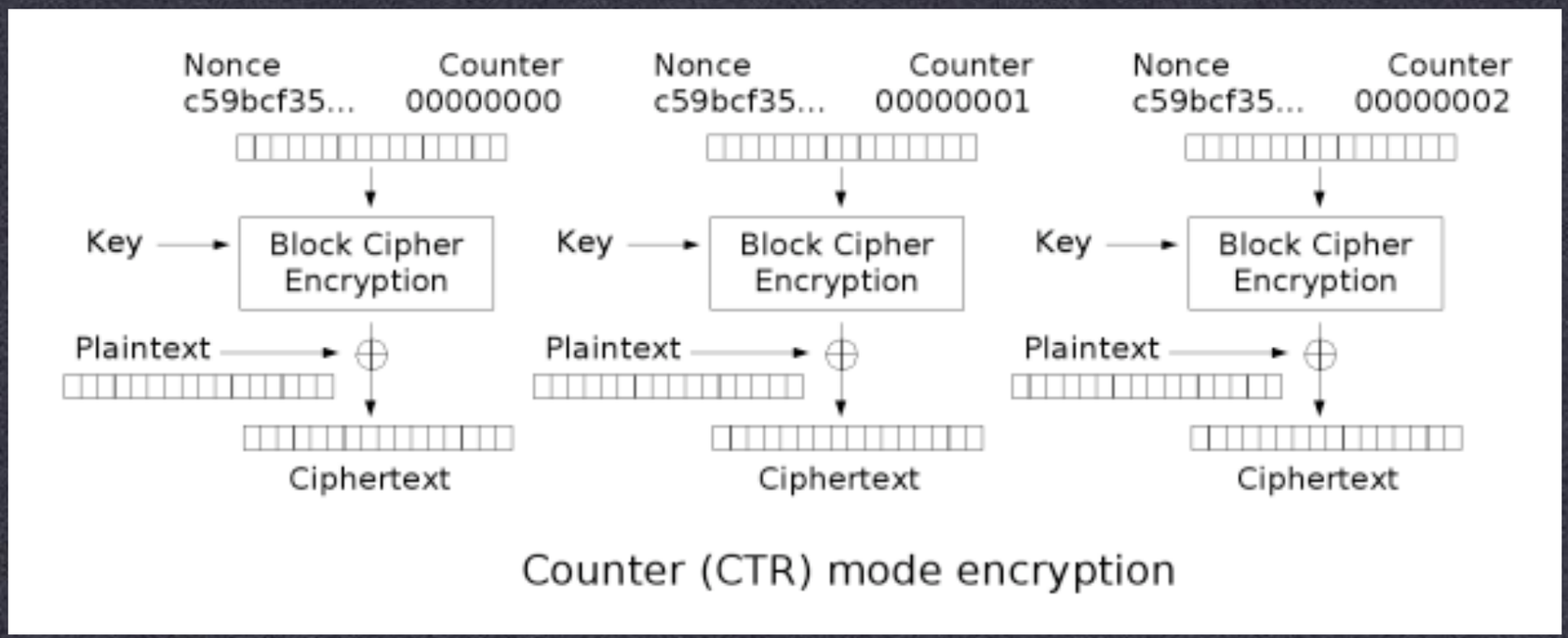
The size of the frame of data to be encrypted or decrypted (i.e. how often a new CBC chain is started) depends on the particular application, and is defined for each in the corresponding format specific books of this specification. Unless otherwise specified, the Initialization Vector used at the beginning of a CBC encryption or decryption chain is a constant, $iv_{0,}$ which is:

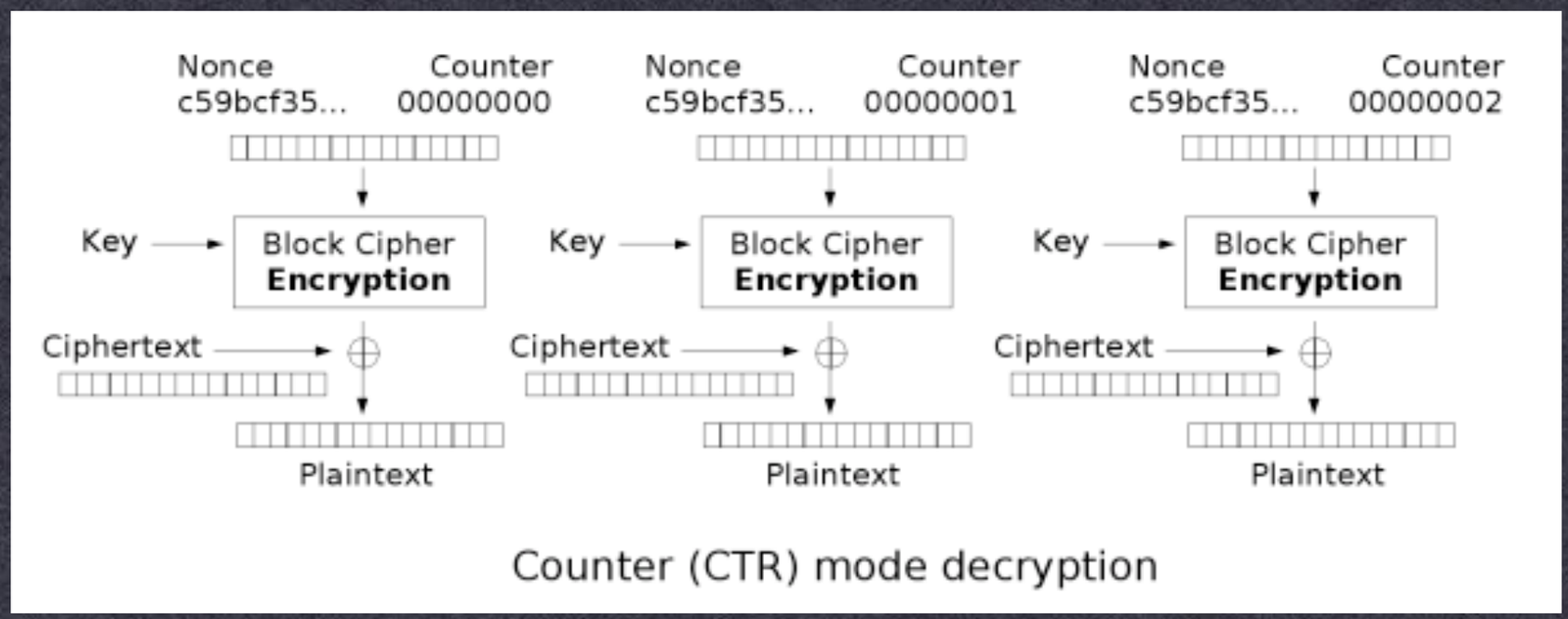
$0BA0F8DDFEA61FB3D8DF9F566A050F78_{16}$

Advanced Access Content System (AACS)

Introduction and
Common Cryptographic Elements

CTR Mode

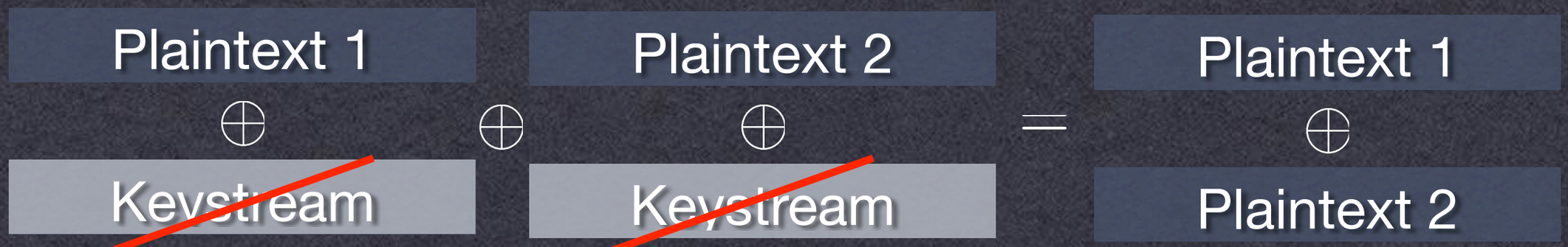




Public domain image courtesy Wikipedia.

Security of CTR

- Yes, assuming secure block cipher (PRP)
- However, counter range must <u>never</u> be re-used



- Similar example: MS Word 2003
  - (they used RC4, but same problem)

Point of order

- **Proofs of security:**
  - **We <u>don't</u> know how to prove that DES or AES are secure block ciphers**
  - **But if we assume that the block ciphers <u>are</u> secure PRPs then:**

-We can prove that CBC & CTR & OFB & CFB etc. are secure encryption modes.

<u>http://www.cs.ucdavis.edu/~rogaway/papers/sym-enc-abstract.html</u>

ChaCha

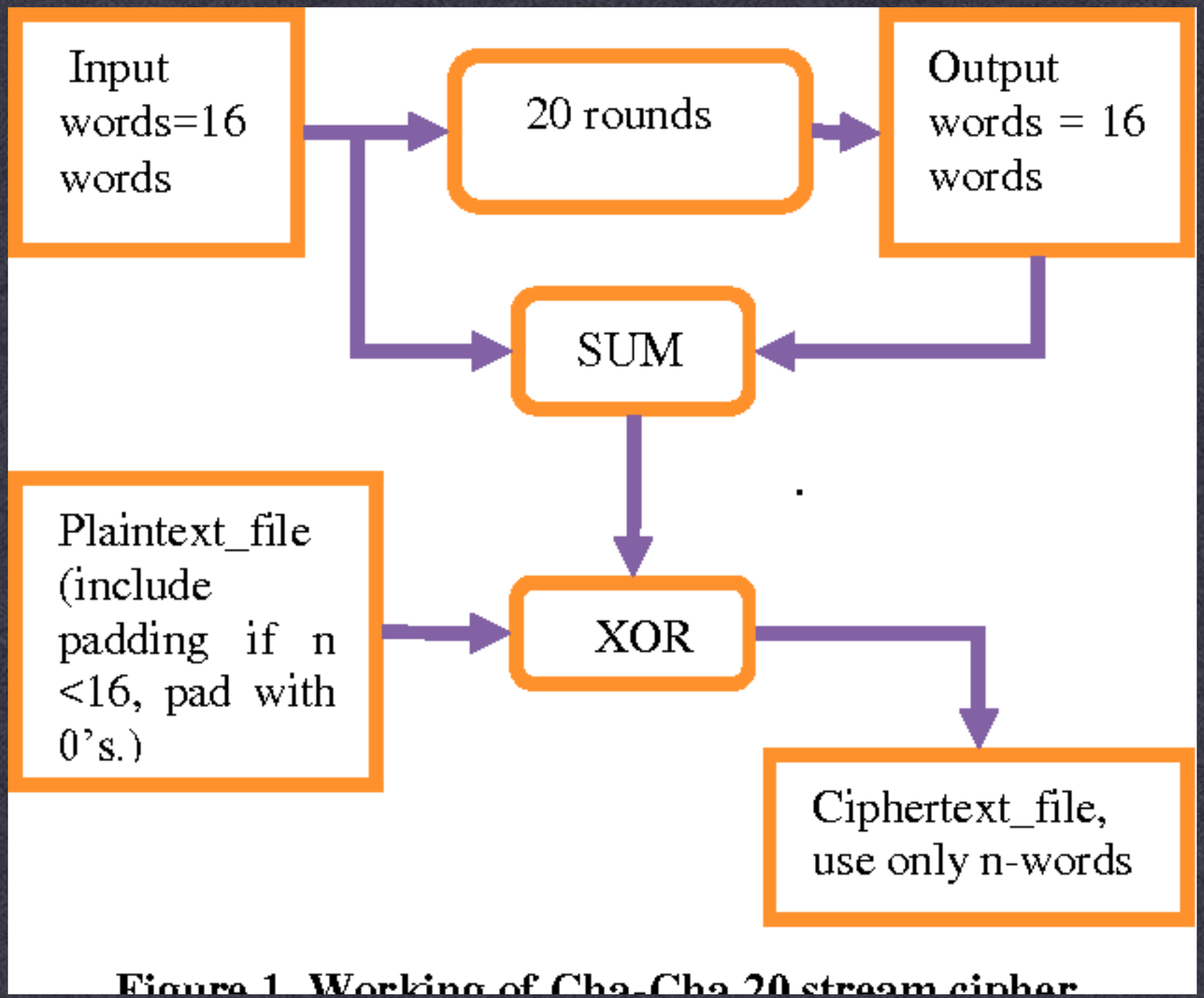
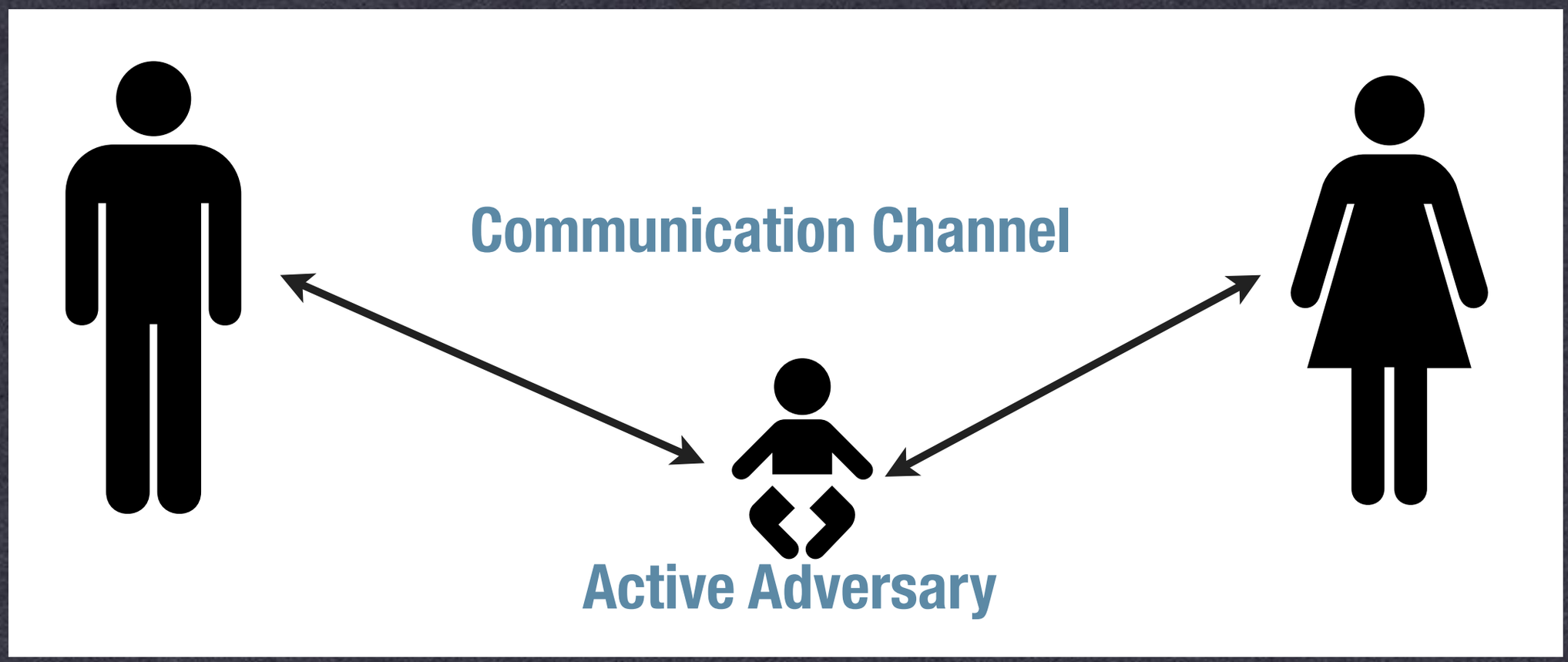


Figure 1. Working of Cha-Cha 20 stream cipher

Malleability

- The ability to modify a ciphertext
  - Such that the plaintext is meaningfully altered
  - CTR Mode (bad)
  - CBC Mode (somewhat bad)

Authenticated Encryption



MACs

- Symmetric-key primitive
  - Given a key and a message, compute a “tag”
  - Tag can be verified using the same key
  - Any changes to the message detectable
- To prevent malleability:
  - Encrypt then MAC
  - Under separate keys

MACs

- Definitions of Security
  - Existential Unforgeability under Chosen Message Attack (EU-CMA)
- Examples:
  - HMAC (based on hash functions)
  - CMAC/CBC-MAC (block ciphers)

Authenticated Encryption

- Two ways to get there:
  - Generic composition
    Encrypt (e.g., CBC mode) then MAC

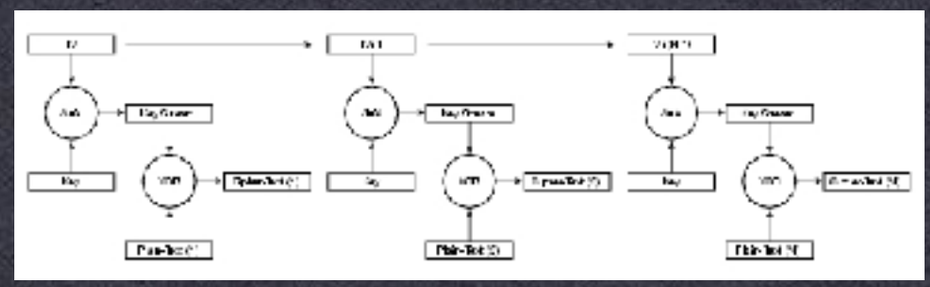
-two different keys, multiple primitives

  - Authenticated mode of operation

-Integrates both encryption & authentication

-Single key, typically uses only one primitive (e.g., block cipher)

-Ex: CCM, OCB, GCM modes



Generic Composition

AEAD

Hash Functions

Asymmetric Crypto

- So far we've discussed <u>symmetric</u> crypto
  - Requires both parties to share a key
  - Key distribution is a hard problem!

Key Agreement

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

D-H Protocol

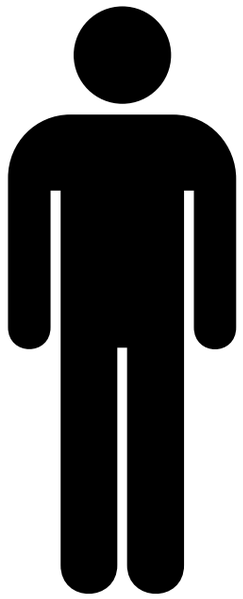
Malcolm Williamson in 72

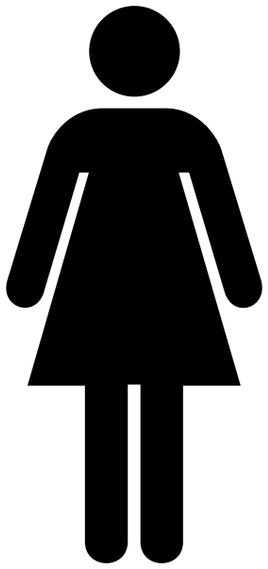
Diffie-Hellman in 76

$b \in \mathbb{Z}_q$

$p, q : p = 2q + 1$

$a \in \mathbb{Z}_q$





$g^a \ mod \ p$

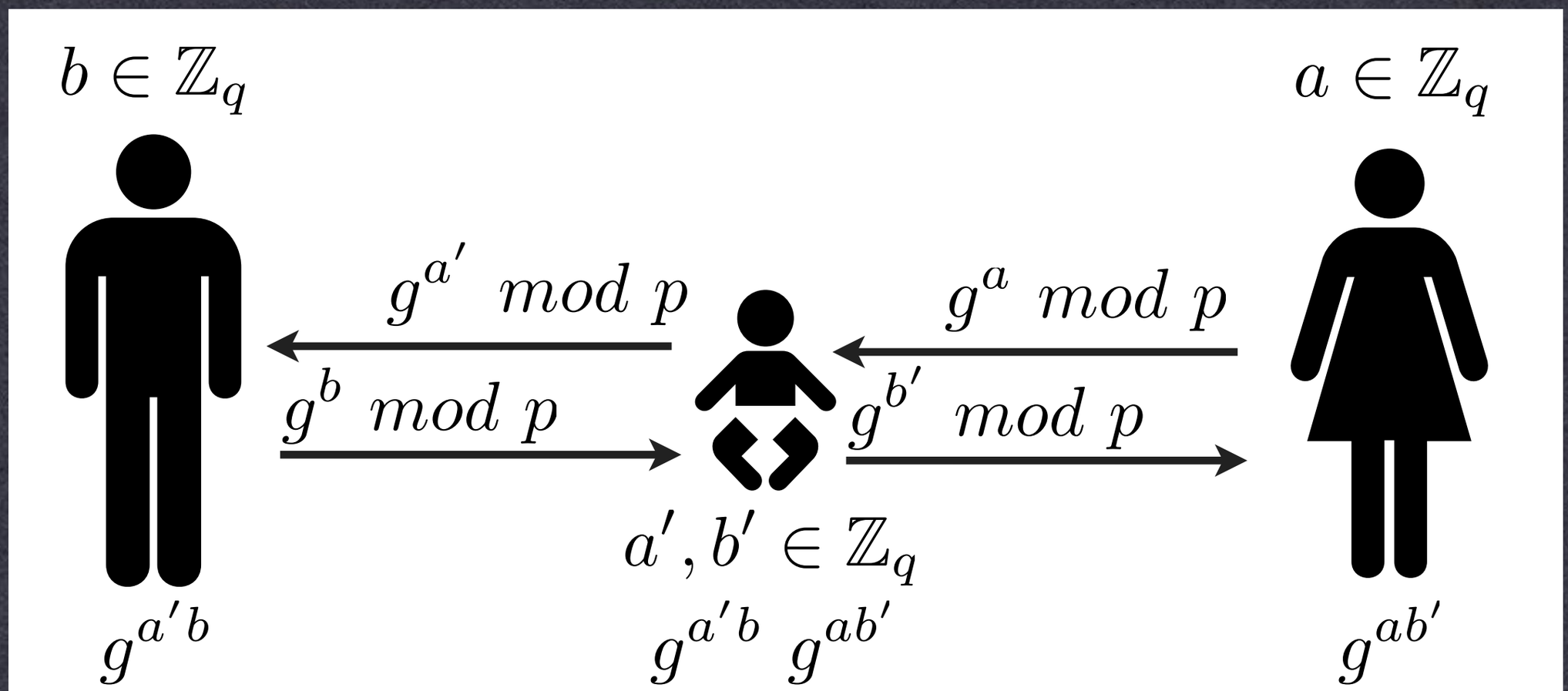
$g^b \ mod \ p$

g^{ab}

g^{ab}

Man in the Middle

- **Assume an <u>active</u> 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...
- Solution: Authenticate the remote party

Preventing MITM

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

Public Key Encryption

- What if our recipient is <u>offline?</u>
  - Key agreement protocols are interac
  - e.g., want to send an email

Ellis in 72, Cocks a few months later

Public Key Encryption

RSA Cryptosystem

Key Generation

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)
  - <u>Malleable</u>

$$c' = c \cdot x^e \ mod \ N$$

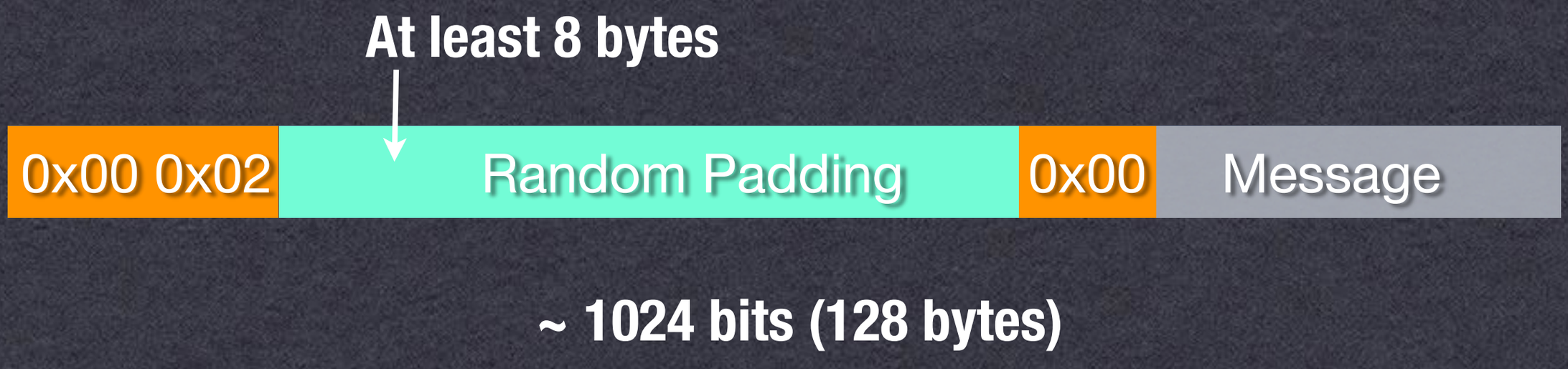
$$c'^d = (m^e \cdot x^e)^d = m \cdot x \ mod \ N$$

  - Might be <u>partially</u> invertible

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

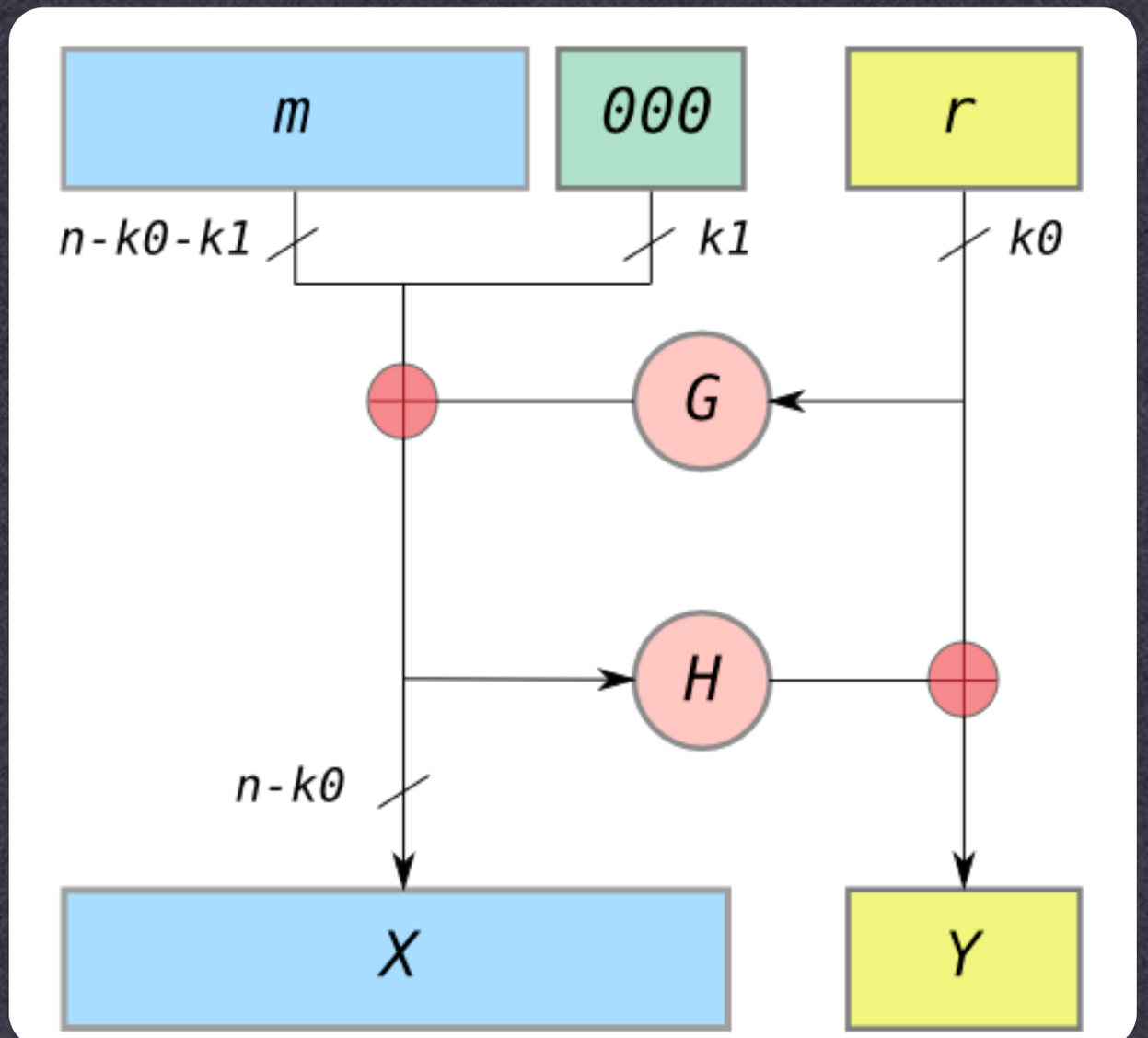
RSA Padding

- Early solution (RSA PKCS #1 v1.5):
  - Add “padding” to the message before encryption
  - Includes randomness
  - Defined structure to mitigate malleability
  - PKCS #1 v1.5 badly broken (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) Encryption	1,016
RSA (1024 bit key) Decryption	21,719

$m^e \ mod \ N$
$e = 65,537$

$m^d \ mod \ N$

Benchmarks from: http://www.cryptopp.com/benchmarks.html
Microsoft Visual C++, Windows XP, Intel Core 2 1.83Mhz in 32-bit mode

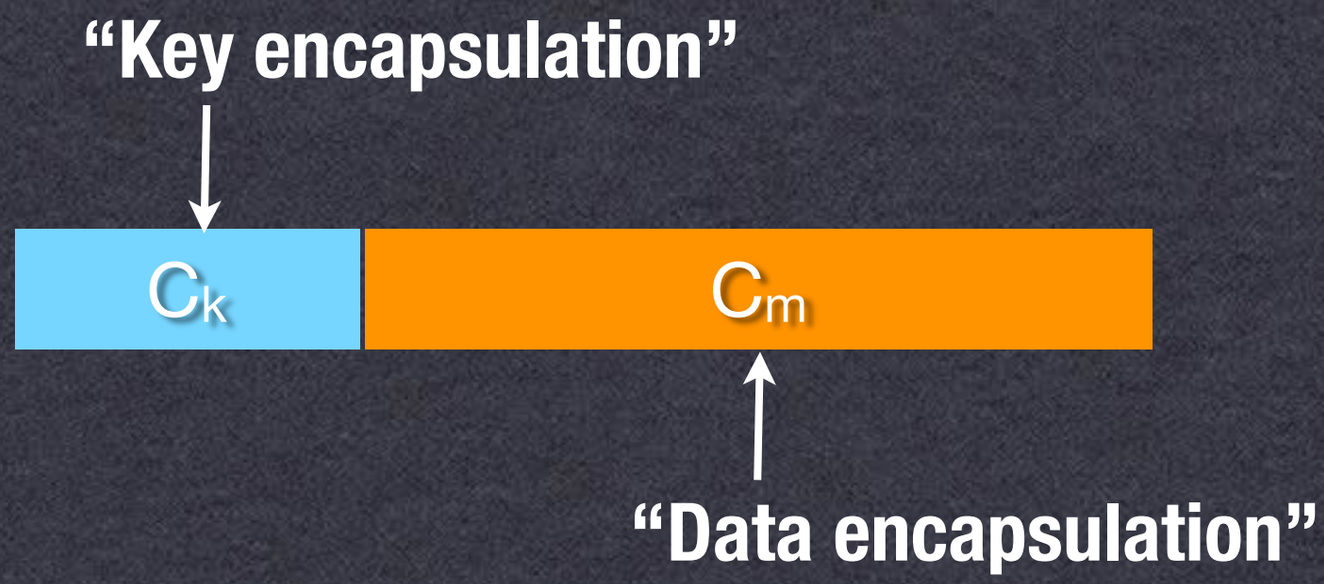
Hybrid Encryption

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

$$k \xleftarrow{\$} \{0,1\}^k$$

$$C_k \leftarrow RSA.Encrypt_{pk}(k)$$

$$C_m \leftarrow AES.Encrypt_k(message)$$



Key Strength

Level	Protection	Symmetric	Asymmetric	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
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

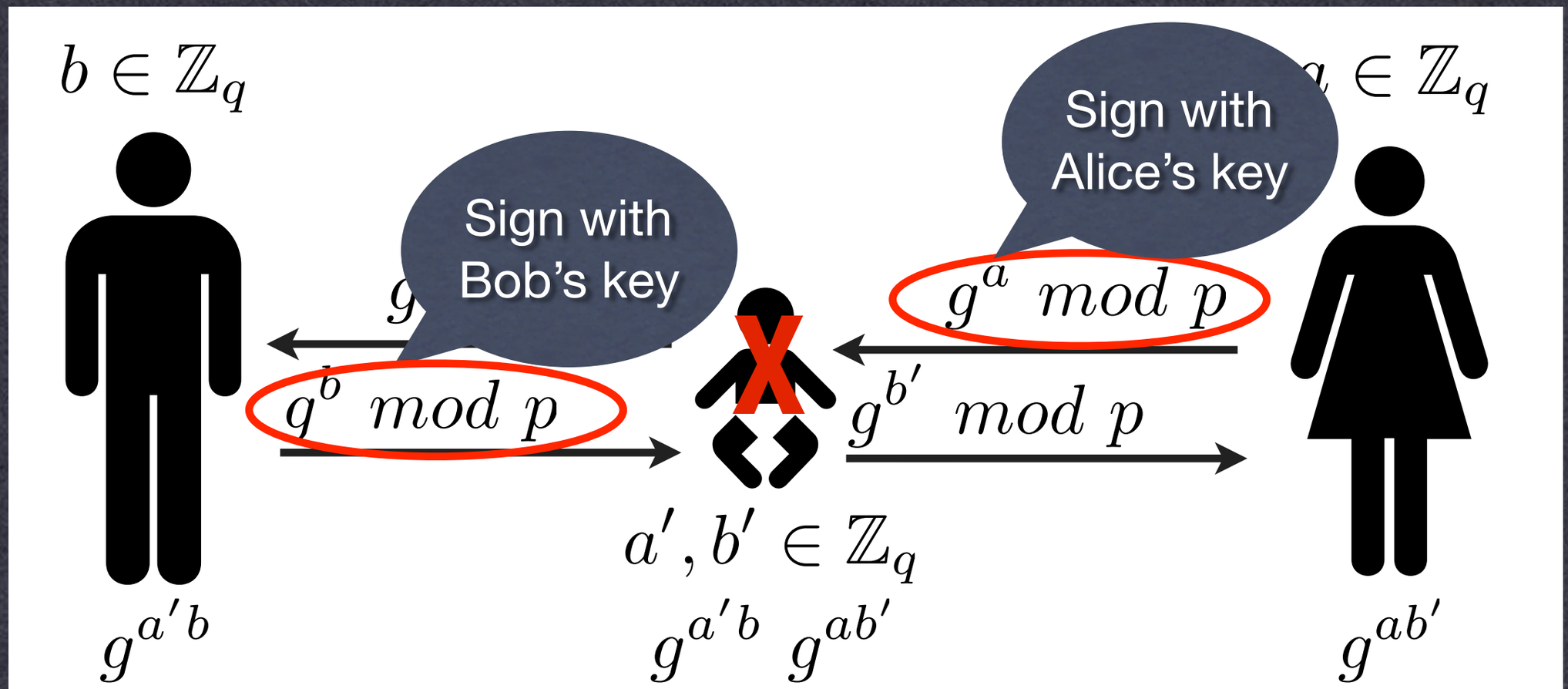
Source: www.keystrength.com (BlueKrypt). Based on ECrypt recommendations.

Digital Signatures

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

Preventing MitM

- **Assume an <u>active</u> adversary:**



PKI & Certificates

- How do I know to trust your public key?
  - Put it into a file with some other info, and get someone else to sign it!

Next Time

- Protocols & Implementation
- Reading!
- A2 coming up this week