Goals of authenticated encryption

Daniel J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

More details, credits:
competitions.cr.yp.to
/features.html

Encryption

$$
\begin{tikzcd}
\boxed{\text{sender}} \arrow[d] & \\
m \arrow[d] & \\
c \arrow[d] & k \arrow[l] \\
\boxed{\text{network}} &
\end{tikzcd}
$$

k: secret key.

m: variable-length plaintext.

c: variable-length ciphertext.

Authenticated encryption

$$\begin{tikzcd}
\boxed{\text{sender}} \arrow[d] & \\
m \arrow[d] & \\
c \arrow[d] & k \arrow[l] \\
\boxed{\text{network}} &
\end{tikzcd}$$

k: secret key.

m: variable-length plaintext.

c: variable-length ciphertext.

Authenticated encryption

$$\begin{tikzcd} \boxed{\text{sender}} \arrow[d] & \\ m \arrow[d] & \\ c \arrow[d] & k \arrow[l] \\ \boxed{\text{network}} & \end{tikzcd}$$

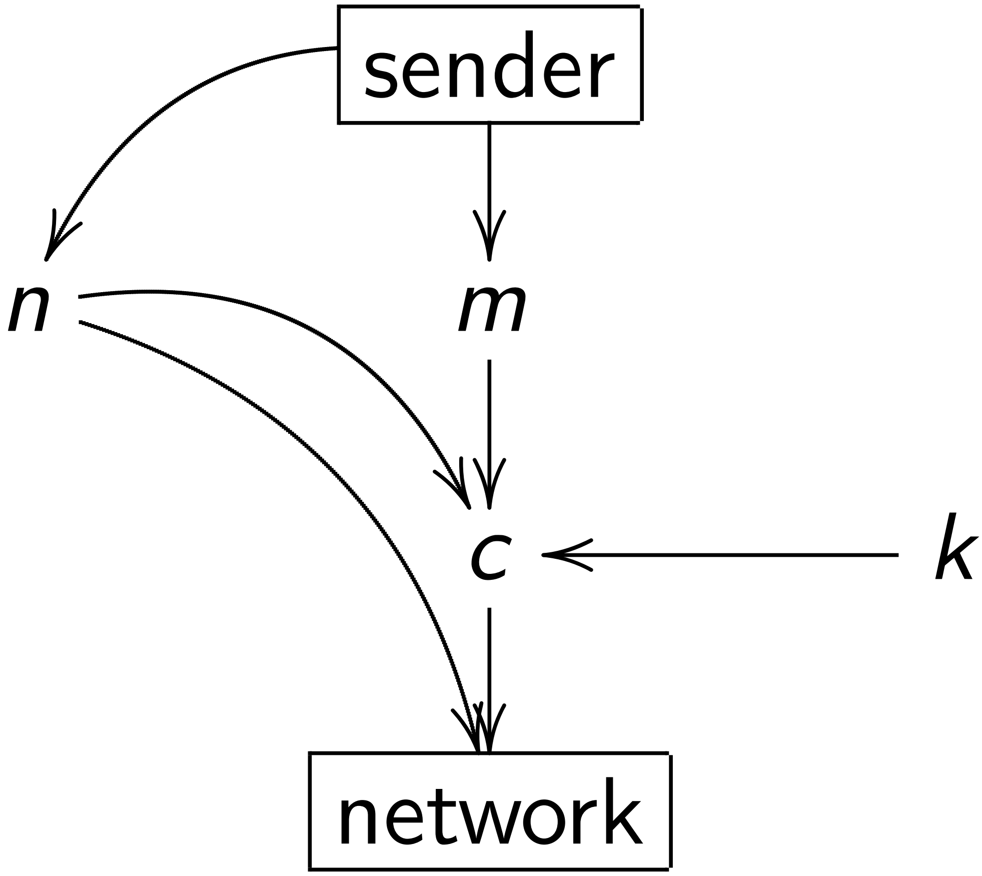
k: secret key.

m: variable-length plaintext.

c: variable-length ciphertext.

Same picture! But now
c is slightly longer than m:
includes an “authentication tag”.

Message numbers



k: secret key.

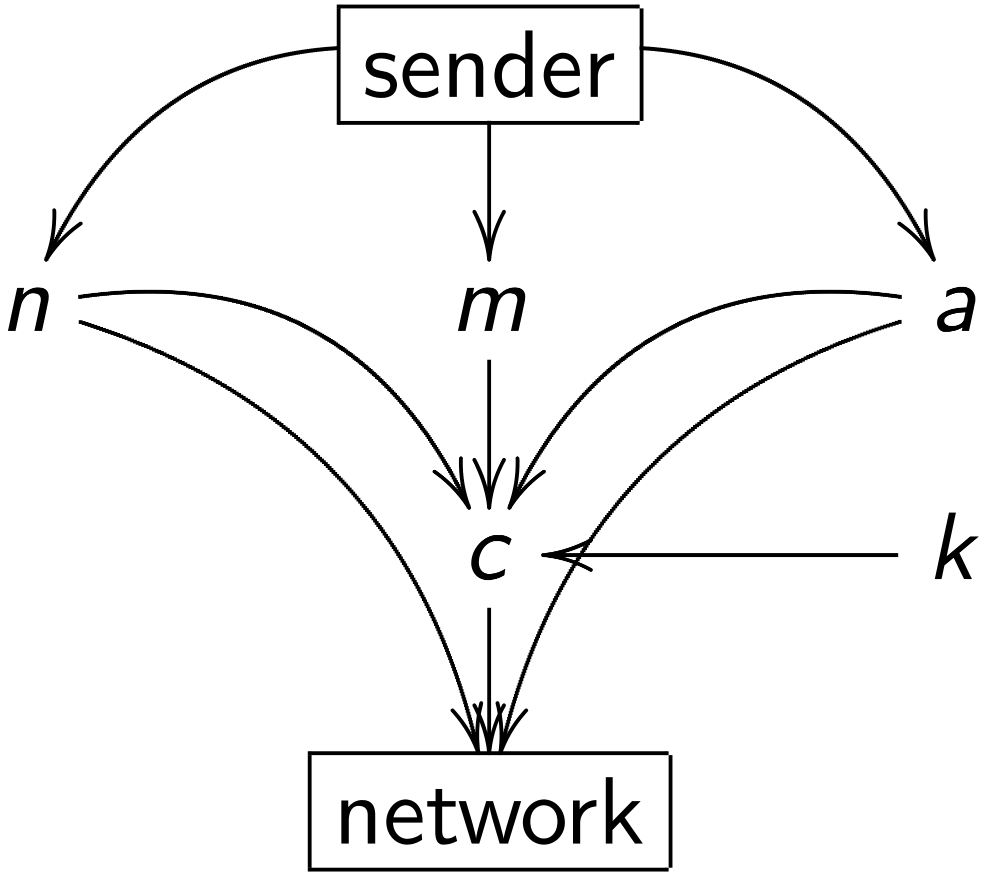
n: public message number.

m: variable-length plaintext.

c: variable-length ciphertext.

Changes in message number hide repetitions of plaintext.

Associated data



k: secret key.

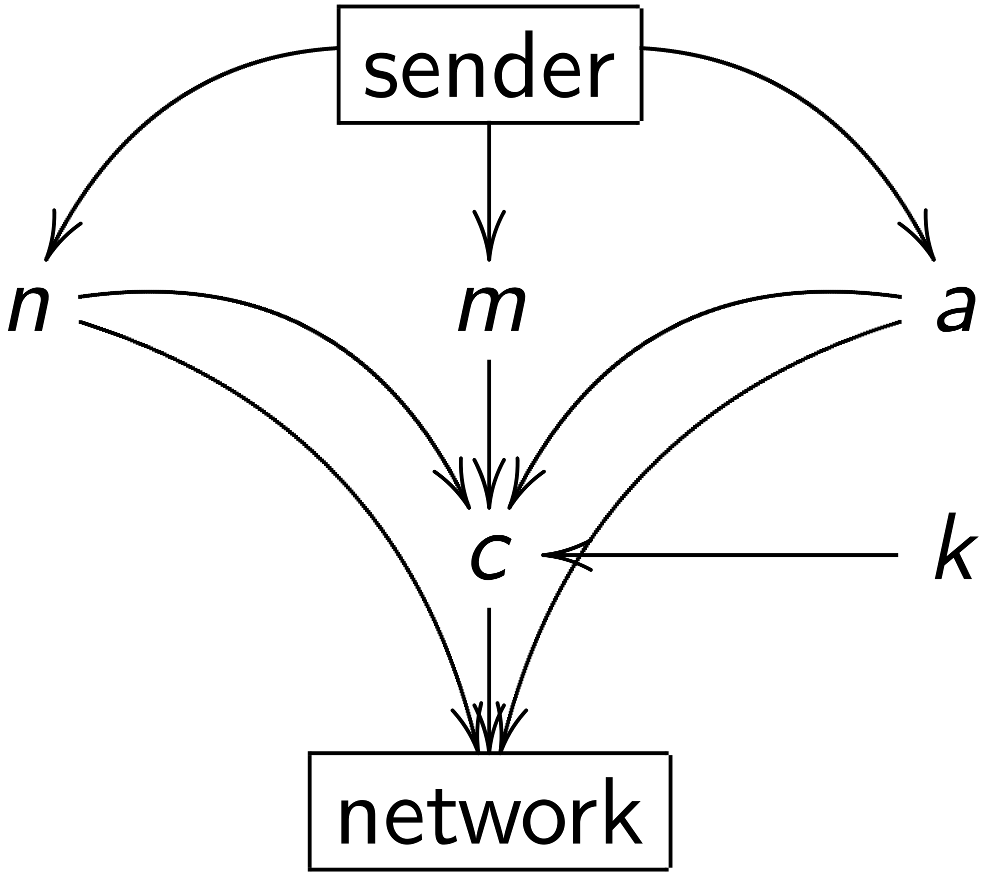
n: public message number.

a: variable-length associated data.

m: variable-length plaintext.

c: variable-length ciphertext.

Associated data



k: secret key.

n: public message number.

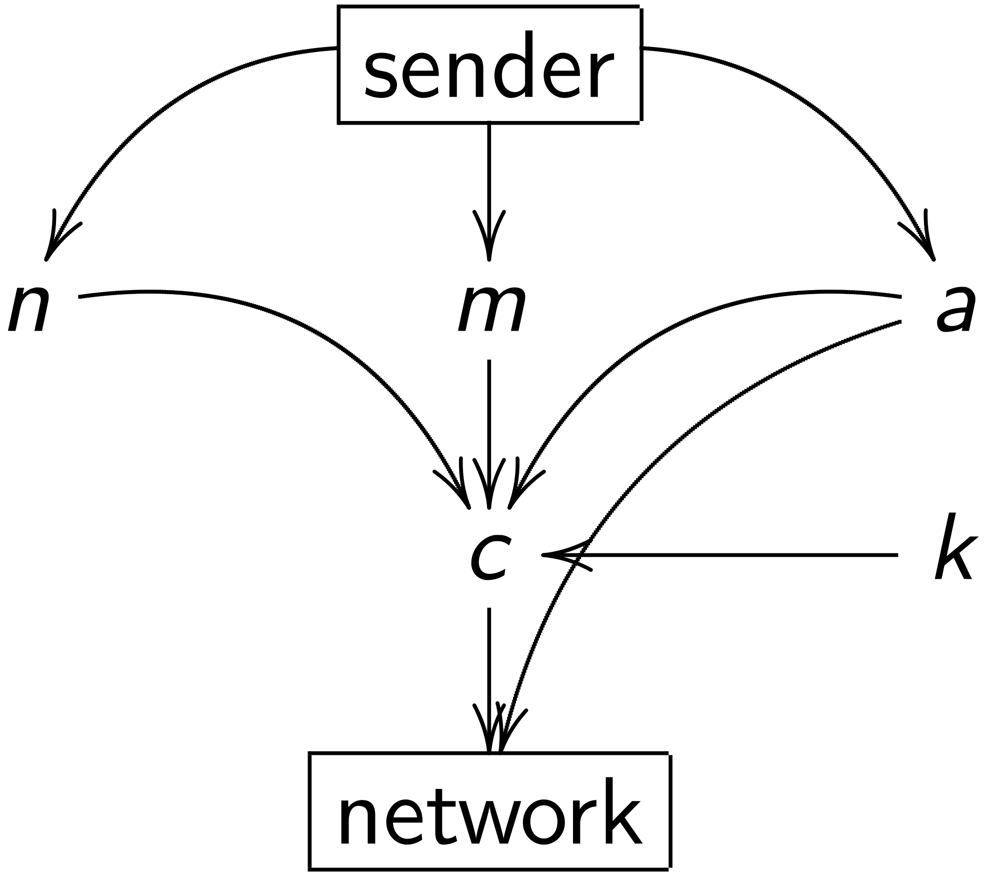
a: variable-length associated data.

m: variable-length plaintext.

c: variable-length ciphertext.

No problem repeating a.

Secret message numbers



k: secret key.

n: secret message number.

a: variable-length associated data.

m: variable-length plaintext.

c: variable-length ciphertext.

What is the attacker's goal?

Plaintext corruption,
associated-data corruption,
message-number corruption.
Forge (n, m, a) that receiver accepts but that legitimate sender never encrypted.

"INT-PTXT" (integrity of plaintexts) means protection against such attacks.

Stronger goal:
Forge at least f messages.

Ciphertext corruption.

Forge c that
receiver accepts but that
legitimate sender never produced.

"INT-CTXT"
(integrity of ciphertexts) means
protection against such attacks.

Ciphertext corruption.
Forge c that
receiver accepts but that
legitimate sender never produced.

"INT-CTXT"
(integrity of ciphertexts) means
protection against such attacks.

Ciphertext prediction.
Distinguish c from
uniform random string.

Ciphertext corruption.
Forge c that
receiver accepts but that
legitimate sender never produced.

"INT-CTXT"
(integrity of ciphertexts) means
protection against such attacks.

Ciphertext prediction.
Distinguish c from
uniform random string.

Is it better to
randomly pad or *zero-pad* a
strong 112-bit MAC to 128 bits?

Replay.
Convince receiver to accept legitimate (n, m, a) more times than legitimate sender sent it.

Replay.
Convince receiver to accept legitimate (n, m, a) more times than legitimate sender sent it.

Reordering.
Convince receiver to accept legitimate messages out of order.

Replay.
Convince receiver to accept legitimate (n, m, a) more times than legitimate sender sent it.

Reordering.
Convince receiver to accept legitimate messages out of order.

Sabotage.
Prevent receiver from seeing (n, m, a) as often as sender sent it: flood radio, switch, CPU, etc.

Replay.
Convince receiver to accept legitimate (n, m, a) more times than legitimate sender sent it.

Reordering.
Convince receiver to accept legitimate messages out of order.

Sabotage.
Prevent receiver from seeing (n, m, a) as often as sender sent it: flood radio, switch, CPU, etc.

Typically delegate solutions to higher-level protocols, but is this optimal?

Plaintext espionage.

Figure out user's secret message.

Plaintext espionage.

Figure out user's secret message.

Message-number espionage.

Figure out user's secret
message number.

Plaintext espionage.
Figure out user's secret message.

Message-number espionage.
Figure out user's secret message number.

Traditional crypto view:
It's okay to use a counter as a message number.
Count is public anyway.

Plaintext espionage.
Figure out user's secret message.

Message-number espionage.
Figure out user's secret message number.

Traditional crypto view:
It's okay to use a counter as a message number.
Count is public anyway.

Counterarguments:
Did attacker see everything?
Maybe timestamp is better, but how much does it leak?
Should encrypt by default.

What are the attacker's resources?

Extensive computation.

Are 80-bit keys adequate?

Are 128-bit keys adequate?

What are the attacker's resources?

Extensive computation.

Are 80-bit keys adequate?

Are 128-bit keys adequate?

Main arguments for small keys:

1. Smaller keys are cheaper.

What are the attacker's resources?

Extensive computation.

Are 80-bit keys adequate?
Are 128-bit keys adequate?

Main arguments for small keys:

1. Smaller keys are cheaper.

2. Attack cost outweighs economic benefit of breaking key, so no sensible attacker will carry out a 2^{80} attack.

Maybe 64-bit keys are enough.

Main counterarguments:

1. Larger keys are cheap enough. User doesn't actually need better performance.

Main counterarguments:

1. Larger keys are cheap enough. User doesn't actually need better performance.

2. Attacker's cost-benefit ratio is improved by multiple-user attacks, multiple forgeries, etc.

Main counterarguments:

1. Larger keys are cheap enough. User doesn't actually need better performance.

2. Attacker's cost-benefit ratio is improved by multiple-user attacks, multiple forgeries, etc.

3. Some attackers carry out attacks that are feasible but not economically rational.

What attacks are feasible?

Back-of-the-envelope figures:

2^{57} watts: received by Earth's atmosphere from the Sun.

2^{44} watts: world power usage.

2^{26} watts: one computer center costing 2^{30} dollars.

1 watt: power for
2^{68} bit operations per year
using mass-market GPUs.

Back-of-the-envelope figures:

2^{57} watts: received by Earth's atmosphere from the Sun.

2^{44} watts: world power usage.

2^{26} watts: one computer center costing 2^{30} dollars.

1 watt: power for
2^{68} bit operations per year
using mass-market GPUs.

Scalable quantum computers.
2^{64} simple quantum operations
to find a 128-bit key
using Grover's algorithm.

Many messages.

Some designers blame the user: "switch keys after 2^{20} messages". Other designers argue that eliminating such requirements adds robustness.

Many messages.
Some designers blame the user: "switch keys after 2^{20} messages". Other designers argue that eliminating such requirements adds robustness.

Chosen plaintexts, chosen ciphertexts, chosen message numbers.
Consensus: Unacceptable to blame the user. All ciphers must be safe against chosen-plaintext attacks and against chosen-ciphertext attacks.

Many users. How does security degrade with number of keys?

Many users. How does security degrade with number of keys?

Repeated message numbers. Minimum impact: Attacker sees whether (n, m, a) is repeated.

Examples of larger impact for many ciphers:

Leak number of shared initial blocks of plaintext.

Leak xor of first non-shared block.

Allow forgery under this n.

Allow forgery under any n'.

Software side channels.

Typical culprits:
secret branches,
secret memory addresses.
Also, on some CPUs,
secret multiplication inputs.

Software side channels.

Typical culprits:
secret branches,
secret memory addresses.
Also, on some CPUs,
secret multiplication inputs.

Hardware side channels.

Power consumption,
electromagnetic radiation, etc.

Software side channels.
Typical culprits:
secret branches,
secret memory addresses.
Also, on some CPUs,
secret multiplication inputs.

Hardware side channels.
Power consumption,
electromagnetic radiation, etc.

Faults. Flip bits in computation.

Software side channels.
Typical culprits:
secret branches,
secret memory addresses.
Also, on some CPUs,
secret multiplication inputs.

Hardware side channels.
Power consumption,
electromagnetic radiation, etc.

Faults. Flip bits in computation.

Thefts and monitors.
Attacker steals secret keys.
Can we still protect
past communication?

What performance is measured?

Typical performance metrics for ASICs:

Low energy (joules) per byte.

Low power (watts).

Low area (square micrometers; loosely predicted by "gate equivalents").

High throughput (bytes per second).

Low latency (seconds; very loosely predicted by cycles).

Similar metrics for
FPGAs and software.

For ASICs and FPGAs,
throughput per se
is not a useful metric
without limit on area (or power).

Parallelize across blocks
or across independent messages
for arbitrarily high throughput.

Similar metrics for
FPGAs and software.

For ASICs and FPGAs,
throughput per se
is not a useful metric
without limit on area (or power).

Parallelize across blocks
or across independent messages
for arbitrarily high throughput.

Fix: measure
ratio of area and throughput, i.e.,
product of area and time per byte.

What operations are measured?

Authenticate only, or encrypt and authenticate?

Cost per byte of a can be far below cost per byte of m.

Send valid data, receive valid data, or receive invalid data?

"Encrypt then MAC" skips cost of decryption for forgeries.

What operations are measured?

Authenticate only, or encrypt and authenticate?

Cost per byte of a can be far below cost per byte of m.

Send valid data, receive valid data, or receive invalid data?

"Encrypt then MAC" skips cost of decryption for forgeries.

Different area targets: encryption/authentication circuit; verification/decryption circuit; circuit that can dynamically select either operation.

Plaintext length and associated-data length.

Huge impact on performance.

Warning: Do not solely compare "overhead" of two ciphers. Cipher with larger "overhead" can be consistently faster.

Plaintext length and associated-data length.
Huge impact on performance.

Warning: Do not solely compare "overhead" of two ciphers. Cipher with larger "overhead" can be consistently faster.

Number of inputs.
e.g. reduce latency by processing several AES-CBC messages in parallel. Simplest if many messages have the same length.

Number of times a key is used.

Most (not all) ciphers
expect precomputation of
"expanded keys".

Warning: Do not solely compare
"agility" of two ciphers.
Cipher with better "agility"
can be consistently slower.

Number of times a key is used.
Most (not all) ciphers expect precomputation of "expanded keys".

Warning: Do not solely compare "agility" of two ciphers. Cipher with better "agility" can be consistently slower.

General input scheduling.
Reduce latency by processing key and nonce before seeing associated data; associated data before plaintext.

Scheduling within plaintext; scheduling within ciphertext.

Typically receive data from left to right. Reduce *latency* by processing earlier parts first.

("Incrementality": Update output efficiently when input is modified.)

Scheduling within plaintext; scheduling within ciphertext.

Typically receive data from left to right. Reduce *latency* by processing earlier parts first.

("Incrementality": Update output efficiently when input is modified.)

Also save *area* if large plaintext does not need large buffer. Warning: Large ciphertext needs large buffer or analysis of security impact of releasing unverified plaintext.

Intermediate tags.

Higher-level protocol splits long plaintext into packets, each separately authenticated. $\Rightarrow$ small buffer is safe.

Do better by integrating similar feature into cipher?

Intermediate tags.
Higher-level protocol splits long plaintext into packets, each separately authenticated.
$\Rightarrow$ small buffer is safe.

Do better by integrating similar feature into cipher?

Other operations.
Single circuit for, e.g., hash and authenticated cipher; for different key sizes; etc.

Intermediate tags.
Higher-level protocol splits long plaintext into packets, each separately authenticated.
$\Rightarrow$ small buffer is safe.

Do better by integrating similar feature into cipher?

Other operations.
Single circuit for, e.g., hash and authenticated cipher; for different key sizes; etc.

Cache context.
How well does the system fit into fast memory?

Support for cryptanalysis

Simplicity.
Cryptanalysts prioritize targets that are easy to understand.

Support for cryptanalysis

Simplicity.
Cryptanalysts prioritize targets that are easy to understand.

Scalability.
Reduced-round targets, reduced-word targets, etc.

Support for cryptanalysis

Simplicity.
Cryptanalysts prioritize targets that are easy to understand.

Scalability.
Reduced-round targets, reduced-word targets, etc.

Proofs.
The phrase "proof of security" is almost always fraudulent.
Proof says that attacks *meeting certain constraints* are difficult, or *as difficult as another problem*.
Can be useful for cryptanalysts.