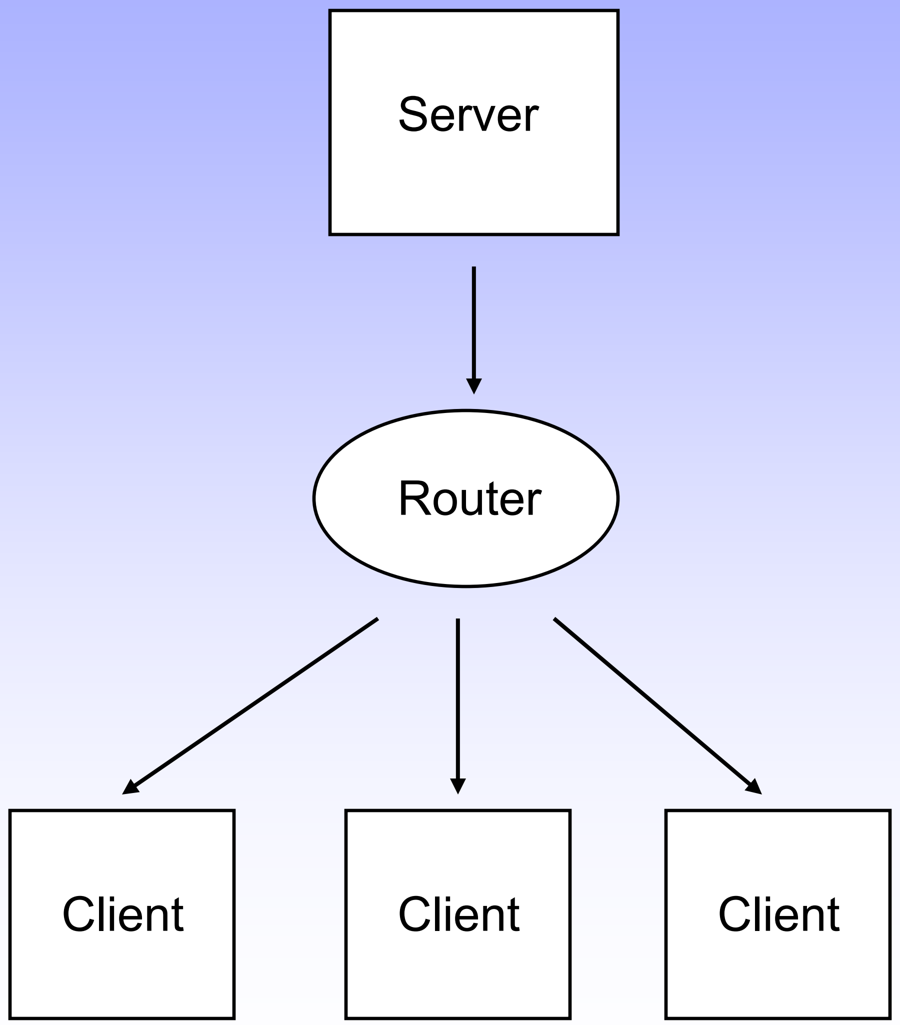
Distillation Codes and DOS Resistant Multicast

Prepared for CS 624 – Fabian Monrose
Johns Hopkins University

Ryan Gardner

Multicast Overview



Multicast Overview

- Multicast enabled routers
- 224.0.0.0 - 239.255.255.255 (class D)
- IGMP (Internet Group Management Protocol)
- Subscribe to groups and unsubscribe

Applications

- Interactive applications
  - Teleconferencing
  - Video conferencing
- Information broadcasts
  - News
  - Stocks
- Updates
  - Software
  - Viruses

Challenges

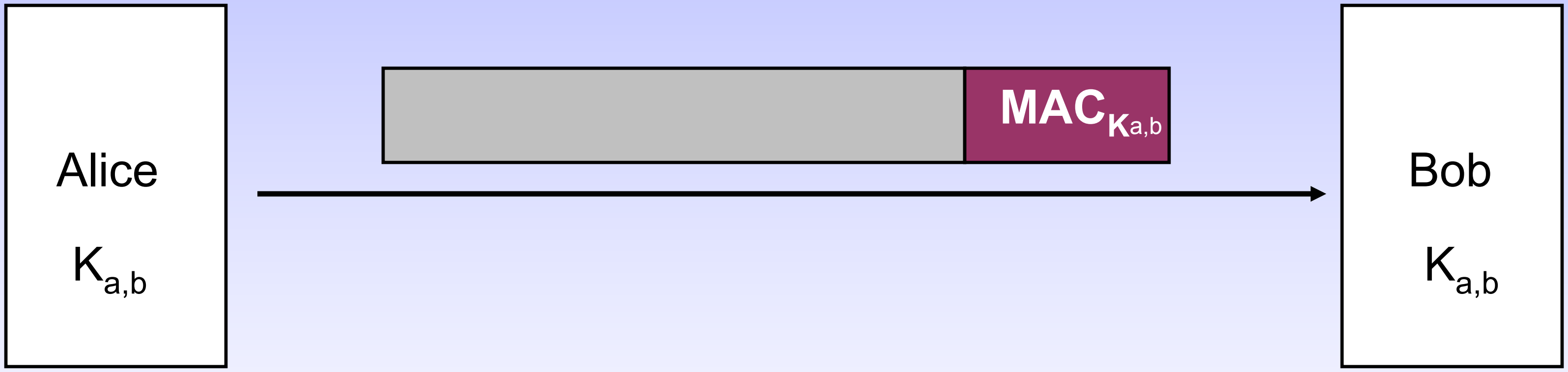
- Authenticity
- Malicious users
- Tolerate packet loss
- Minimal delay
- (DoS attacks)

Outline

- Three naive solutions
- Brief summary of related work
- Efficient Multicast Stream Authentication using Erasure Codes
- Distillation codes
- Conclusion

Naive Solution 1
Symmetric Authentication

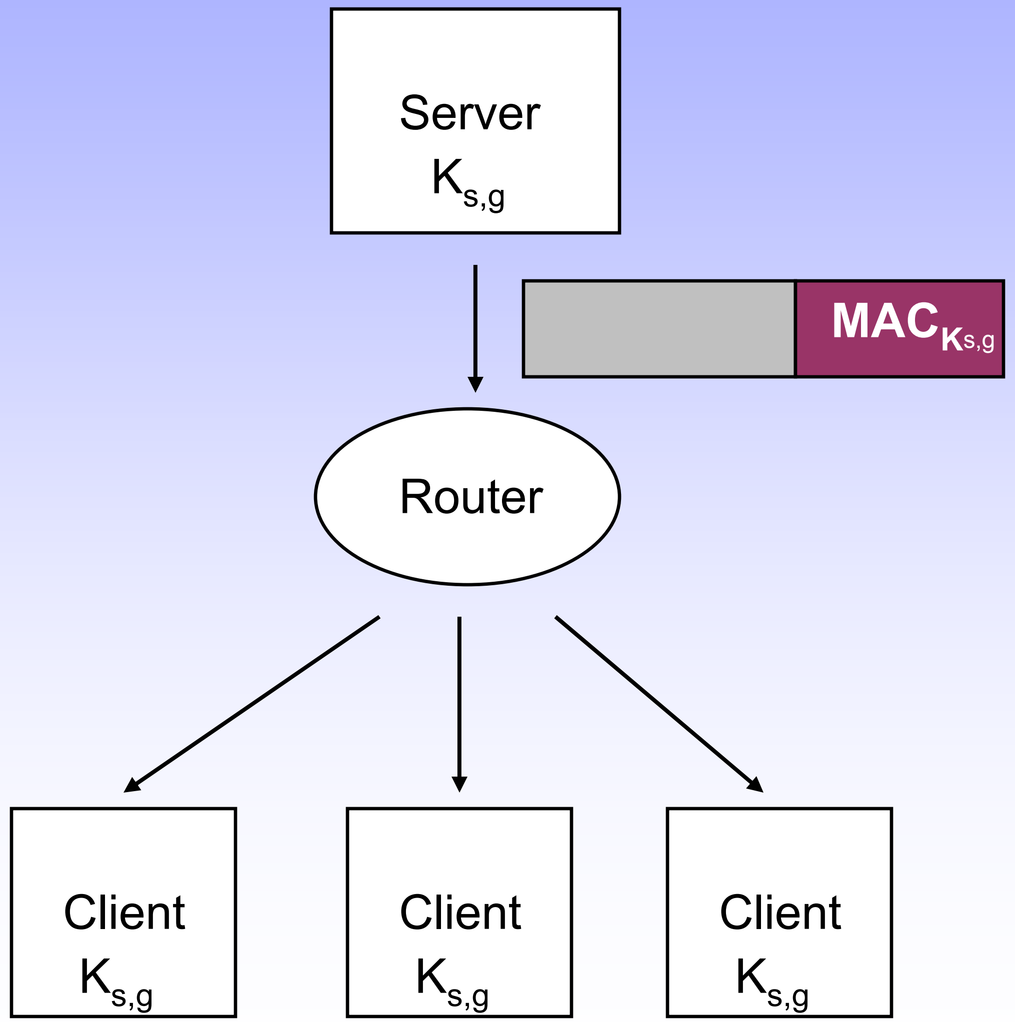
Review of MAC



Examples

- hmac (sha1, md5)
- umac
- cbc mac (aes, 3des)

Naive Solution 1
Symmetric Authentication

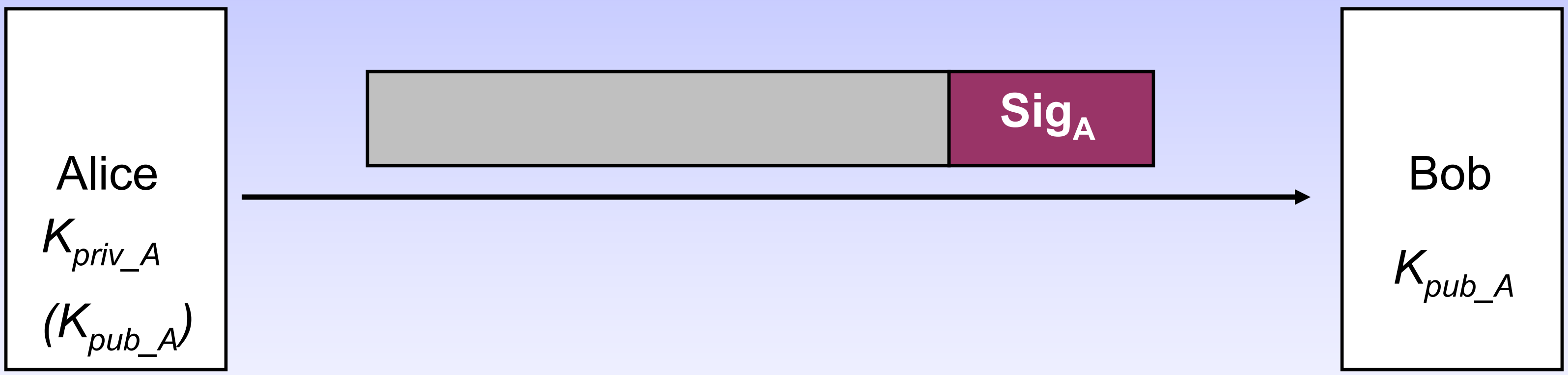


Naive Solution 1
Symmetric Authentication

- Pros
  - Fast
  - Low space overhead
  - Virtually no delay
  - Simple
- Cons
  - Any member of the group can “authenticate packets”

Naive Solution 2
Sign Every Packet

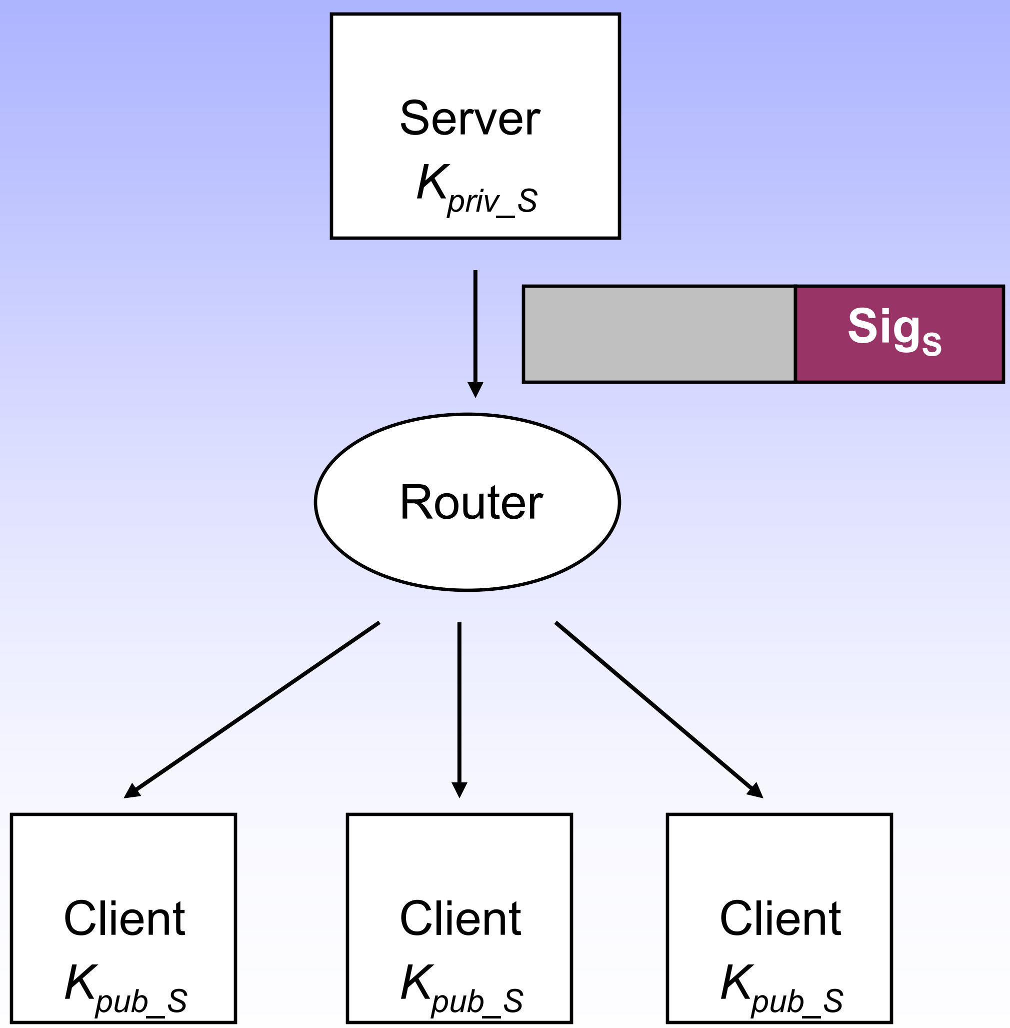
Review of Signature



Examples

- RSA-1024 (2048, etc.)
- DSA
- IBE short signatures

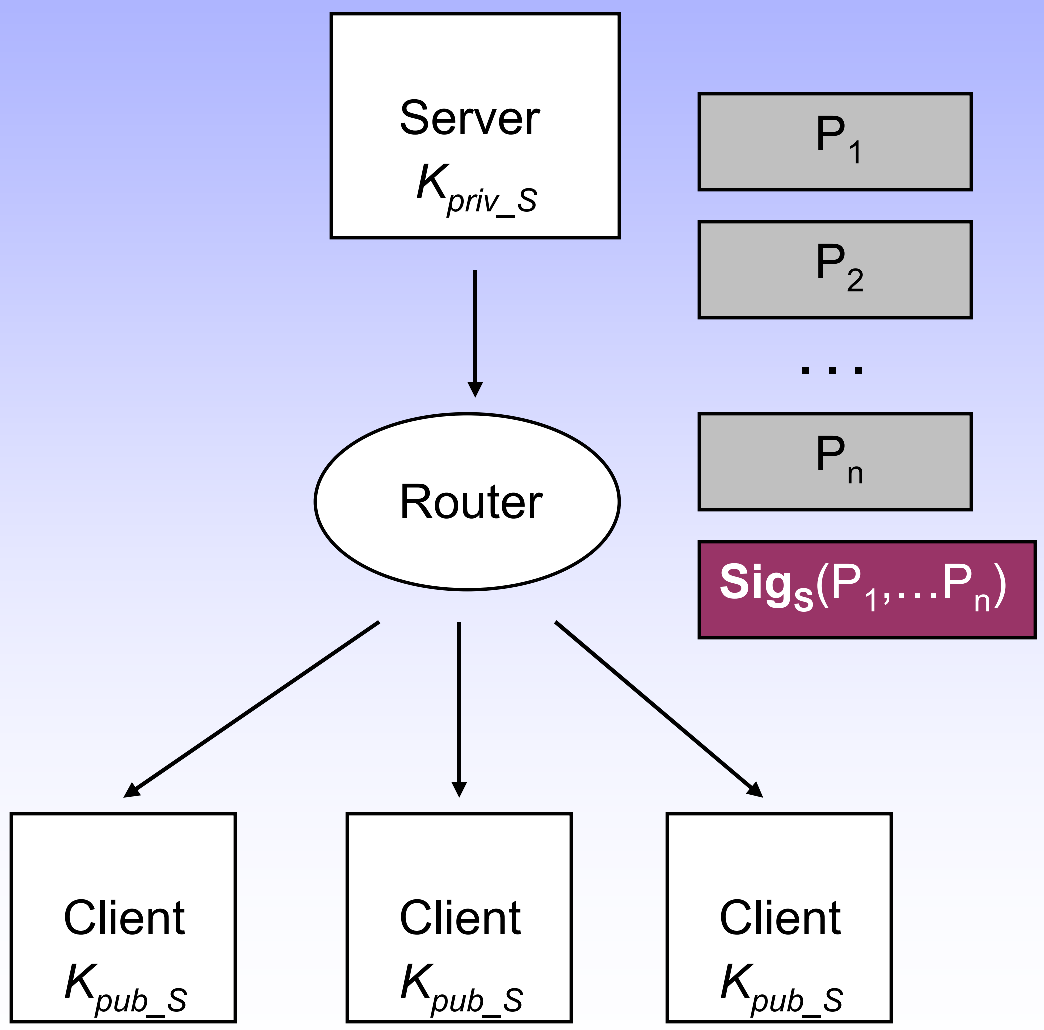
Naive Solution 2
Sign Every Packet



Naive Solution 2
Sign Every Packet

- Pros
  - Guarantees authenticity
  - Perfect loss tolerance
  - Almost no delay
- Cons
  - Computationally expensive for sender and receiver
  - High bandwidth overhead

Naive Solution 3
Basic Signature Amortization



Naive Solution 3
Basic Signature Amortization

- Pros
  - Unforgeable
  - Low computational cost
  - Low bandwidth overhead
- Cons
  - *No* packet loss tolerance
  - Delay at receiver

Outline

- Three naive solutions
- Brief summary of related work
- Efficient Multicast Stream Authentication using Erasure Codes
- Distillation codes
- Conclusion

Related Work

- "Asymmetric MACs"
  - TESLA [12,13]
  - Biba "signature" [11]
- Signature amortization…

Signature Amortization

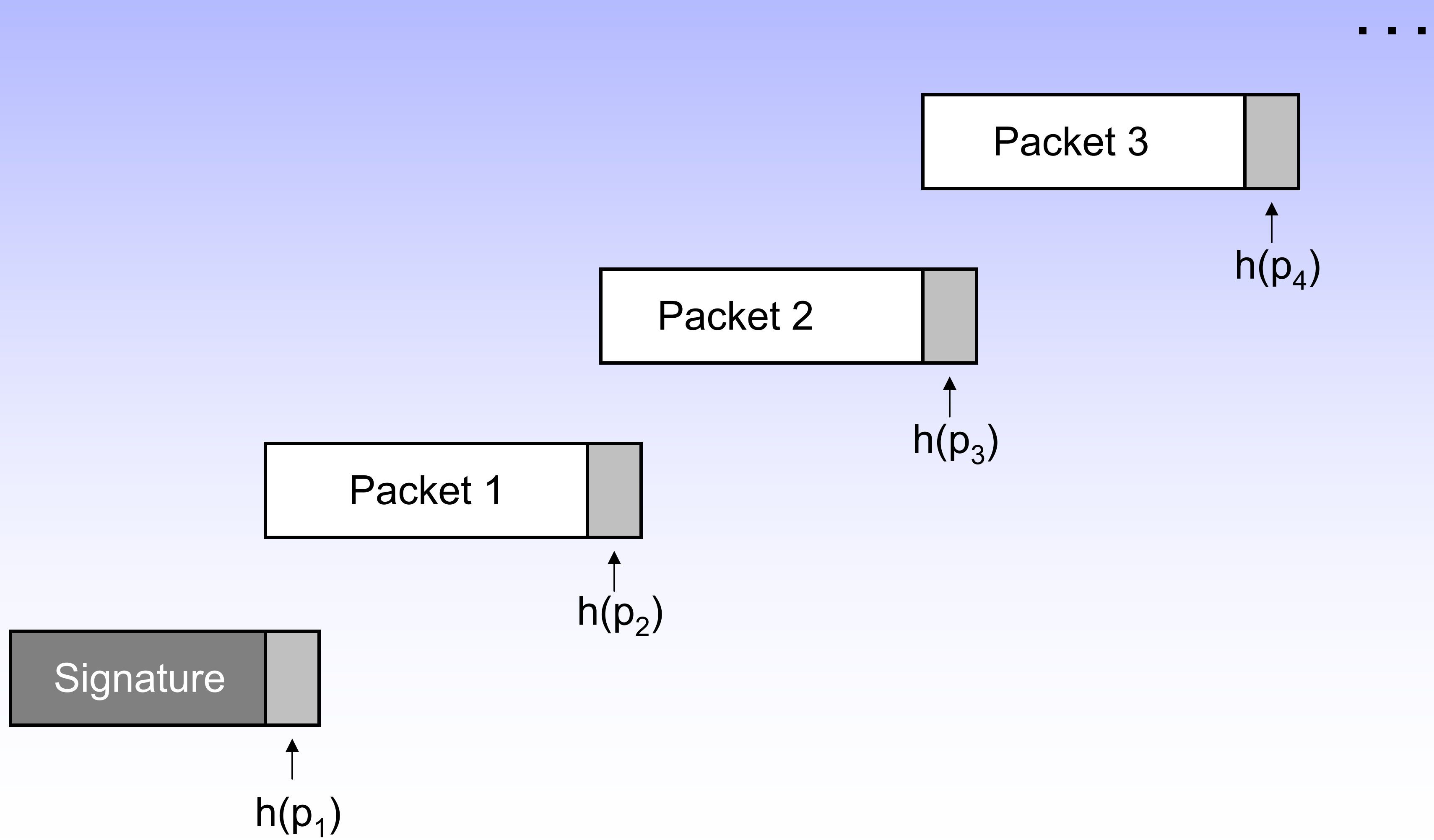
- Signature generations are expensive
- Boneh, Durfee, and Franklin showed can't use MACs entirely… [2]
- Break single signature into multiple packets
- Fundamental issues
  - Packet loss
  - Maliciously inserted packets (DoS)
- Some work done
  - Accumulators [16]
  - Erasure Codes [9,10]

How to Sign Digital Streams [4]

***CRYPTO* '97**

- Objectives
  - Stream signing (not necessarily multicast)
  - Authenticity
  - Non-repudiation (even for partial streams)
  - Inexpensive
  - Low delay
- General approach
  - Authentication chain bootstrapped with signature

How to Sign Digital Streams



How to Sign Digital Streams

- Pros
  - Simple
  - Low computation (single signature)
  - Low overhead
  - Authenticity
  - Non-repudiation (even for partial streams)
  - Low delay (if packets are sent at high frequency)
- Cons
  - No loss tolerance

Digital Signatures for Flows and Multicasts [16]

IEEE/ACM Transactions on Networking 1999

- Objectives
  - Authenticity
  - “High” signing and verification rates
  - Loss tolerant
  - Non-repudiation
  - Inexpensive
  - Low delay
- General approach
  - Create a common signature for blocks of packets
  - Self authenticating packets

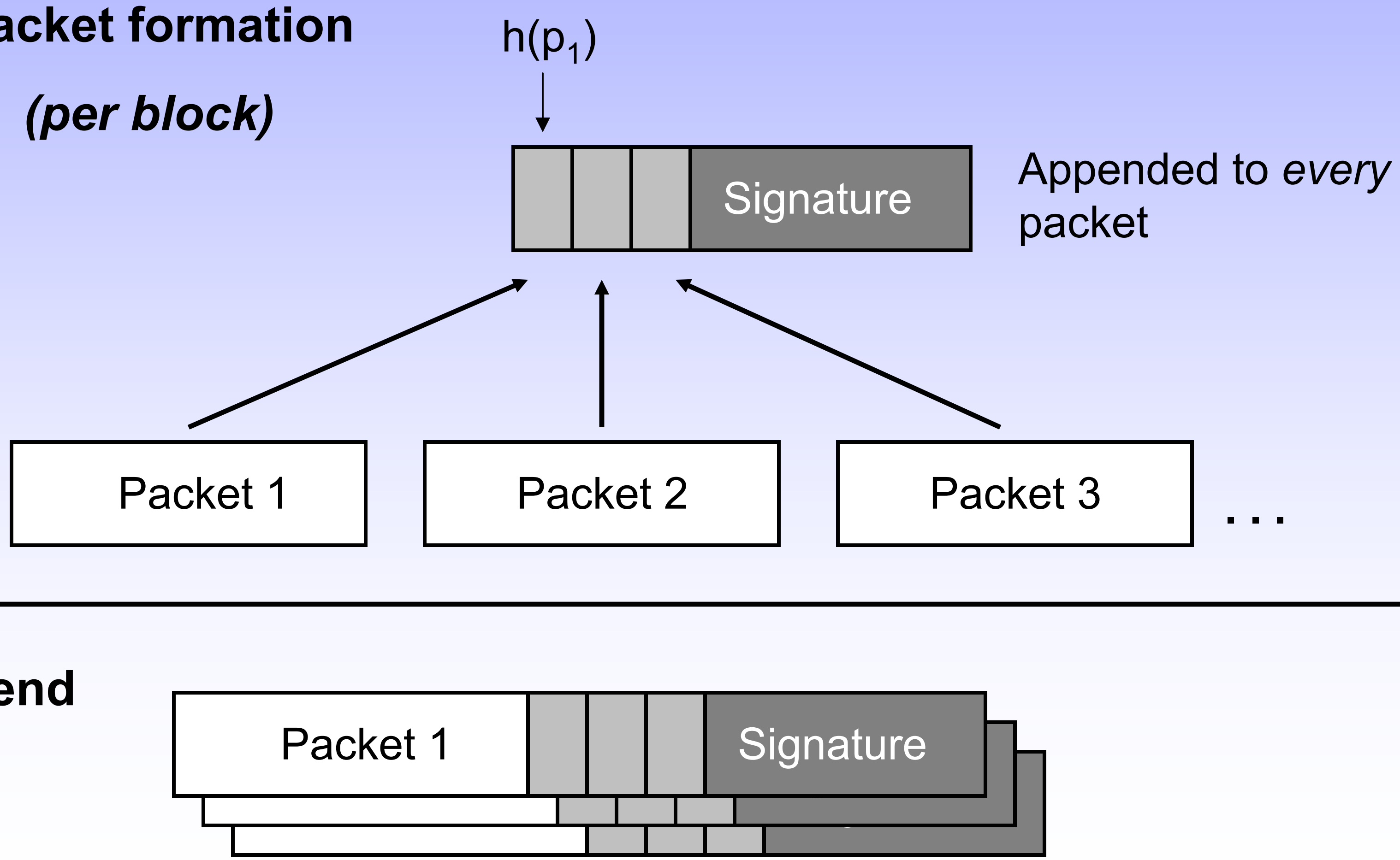
Digital Signatures for Flows and Multicasts

Star Chaining

Packet formation

(per block)

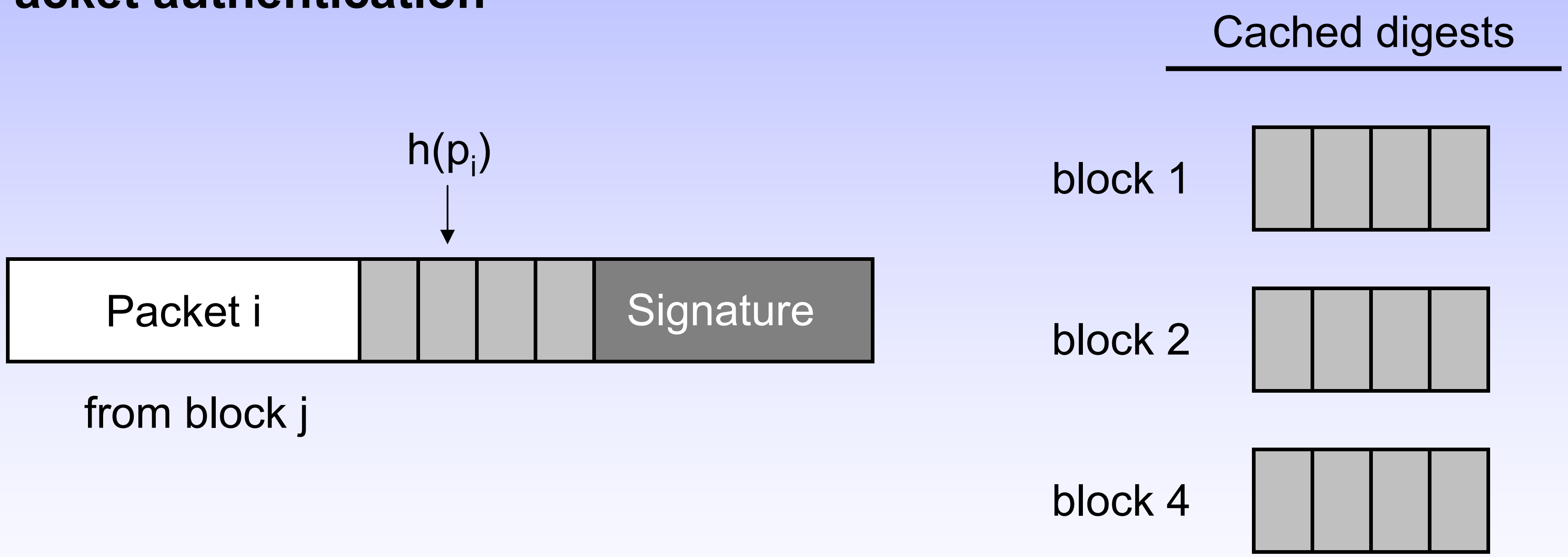
Send



Digital Signatures for Flows and Multicasts

Star Chaining

Packet authentication



Digital Signatures for Flows and Multicasts

- Pros
  - Authenticity
  - “High” signing and verification rates
  - Perfect loss tolerance
  - Non-repudiation
- Cons
  - Small sender delay
  - Extremely high bandwidth overhead

Summary of Related Work

- Still significant deficiencies
  - No loss tolerance
  - Extremely high bandwidth overhead
  - Vulnerable to DoS attacks
    - Computational
    - Memory exhaustion

Outline

- Three naive solutions
- Brief summary of related work
- Efficient Multicast Stream Authentication using Erasure Codes
- Distillation codes
- Conclusion

Efficient Multicast Stream Authentication using Erasure Codes [10]

ACM Transactions on Information and Systems Security 2003

- Objectives
  - Ensure authenticity (non-repudiation)
  - Robustness to packet loss
  - Minimal overhead & delay
  - Robust against en route packet modification or insertion of small number of bogus packets
- General approach
  - Amortize a signature over several packets using erasure codes

Erasure Codes

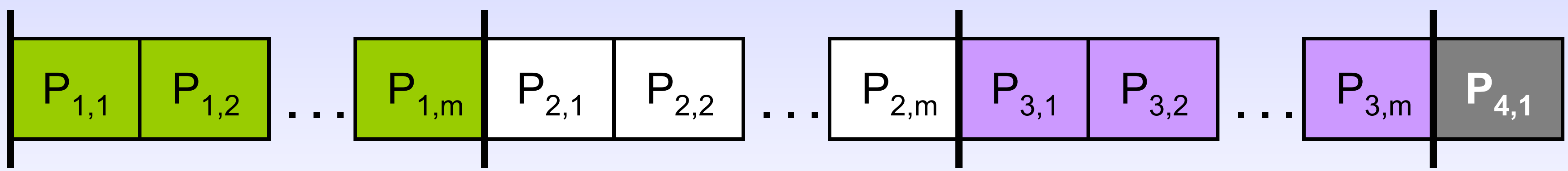
- Sender
  - Take *m* objects (the original data) and creates *n* “erasure encoded objects”
- Receiver
  - Needs any *m* of the *n* objects sent, and can reconstruct “erasure decode” the original data
- Space optimal

Information Dispersal Algorithm (IDA) [14]

- Basics
  - Create an *n* row matrix A such that any *m* of the *n* rows are linearly independent
  - Multiply that by our data
  - On receipt of m chunks, grab the corresponding *m* rows of A, A’
  - Multiply received data by A'^{-1}
- Kevin will cover…
- Pretty light computationally
  - One matrix multiplication at each end (matrix inversion at receiver)
  - $O(n^2)$ encode
  - $O(m^2)$ decode

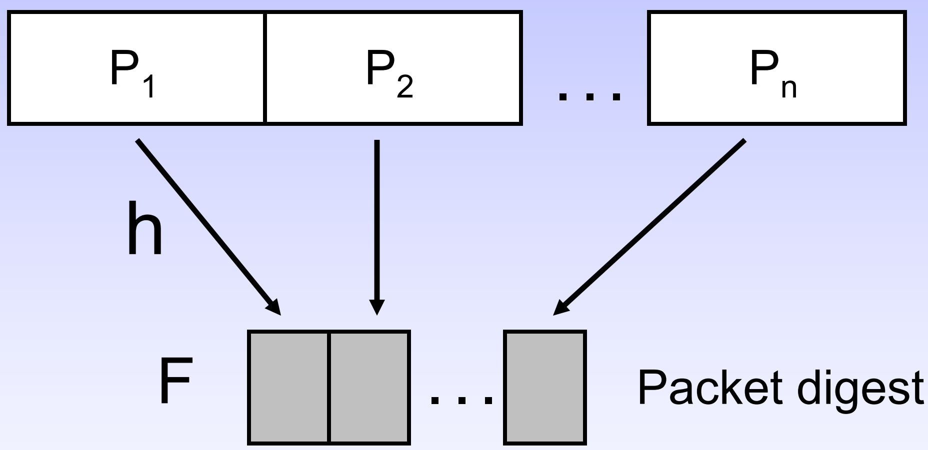
Signature Amortization using IDA - Description

Break a stream up into blocks



Signature Amortization using IDA

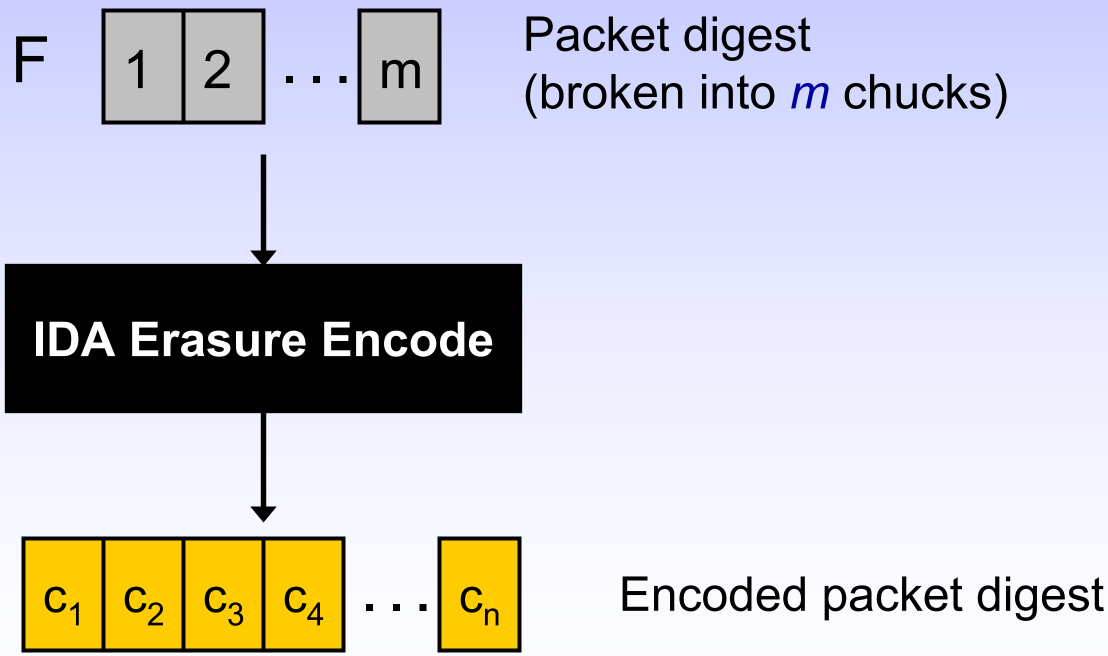
For each block



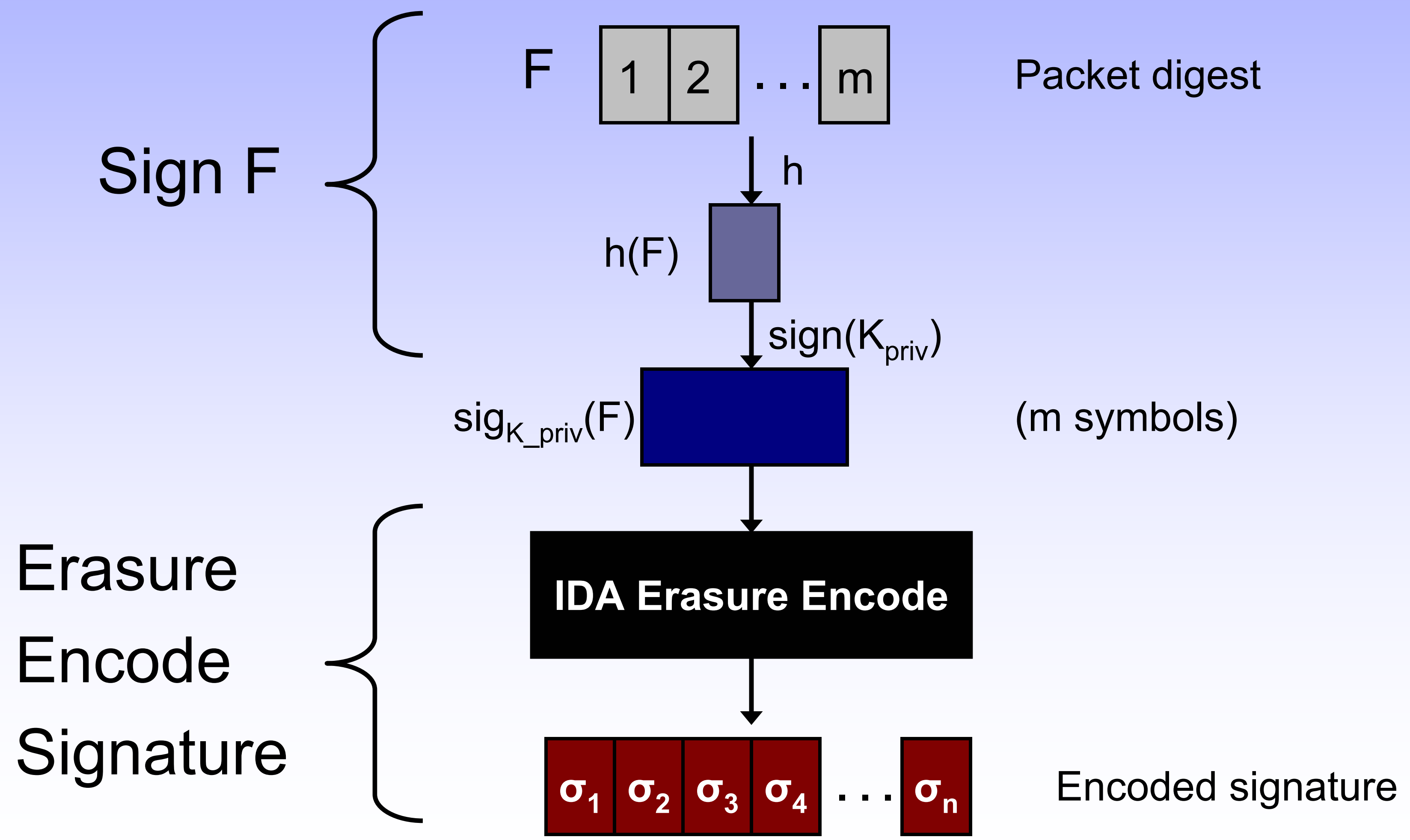
$$F = h(P_1)||\ h(P_2)||\ \ldots\ ||h(P_n)$$

Signature Amortization using IDA

Erasure encode F using IDA

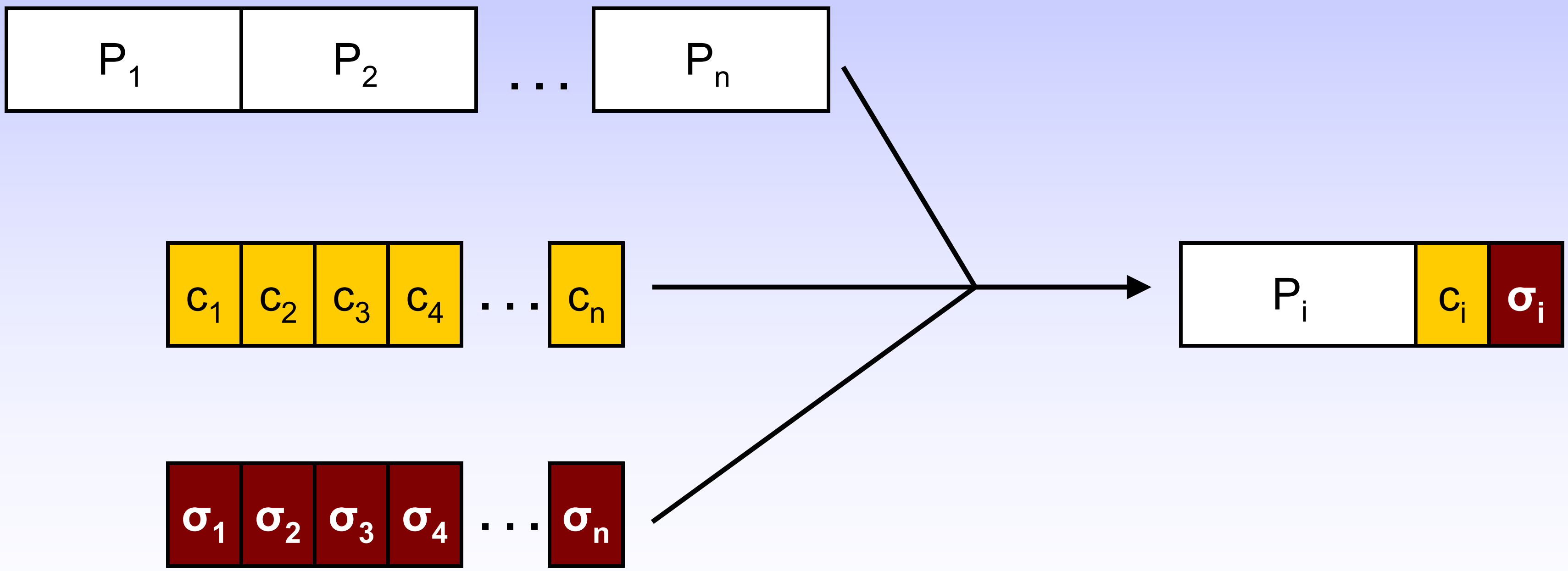


Signature Amortization using IDA



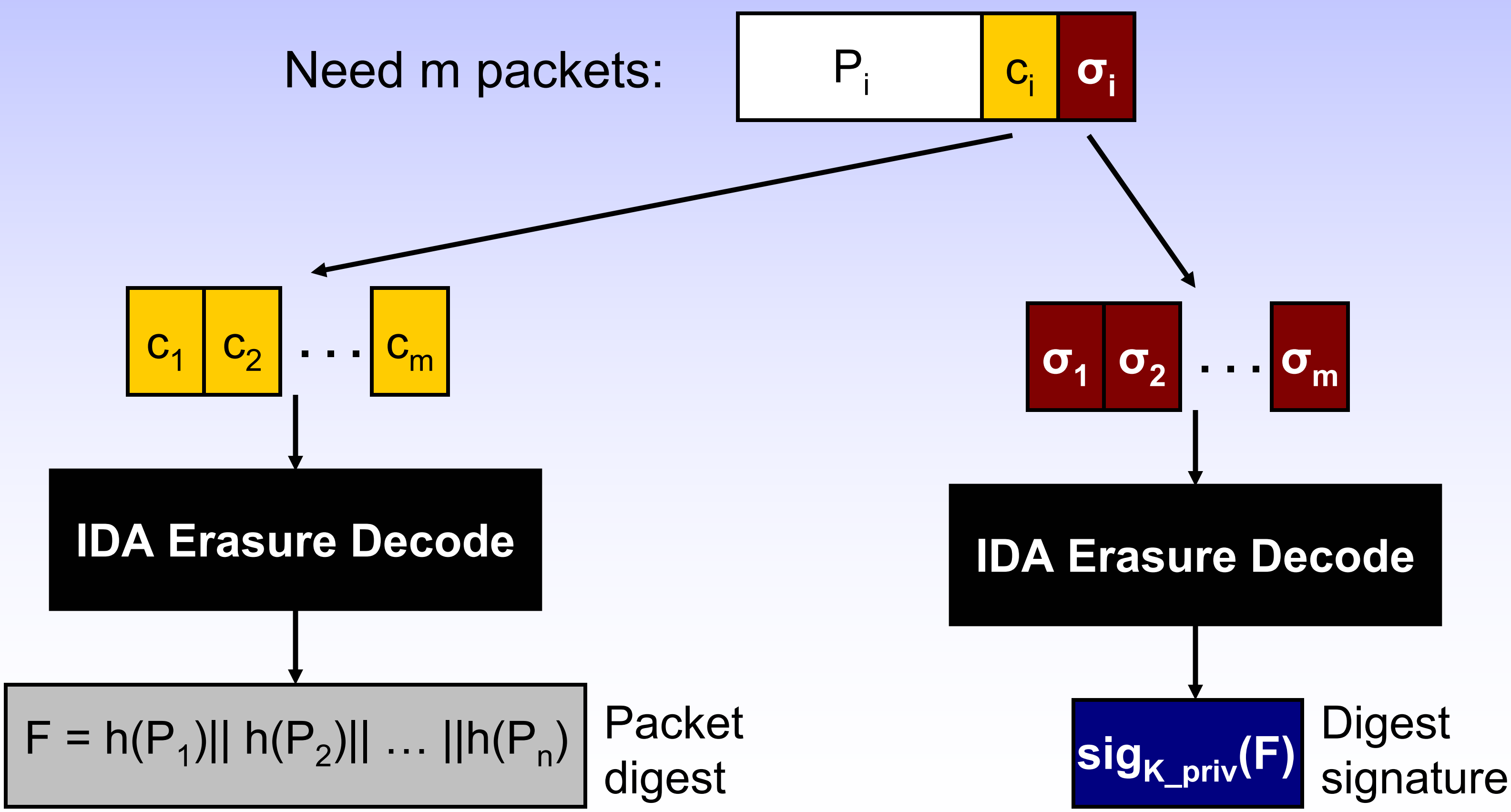
Signature Amortization using IDA

Form each packet



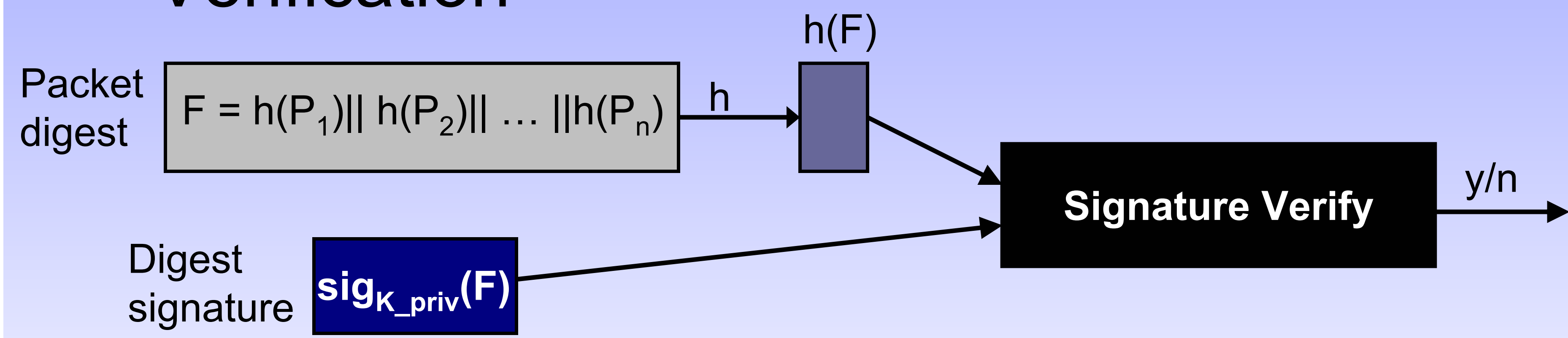
Signature Amortization using IDA

Reconstruction

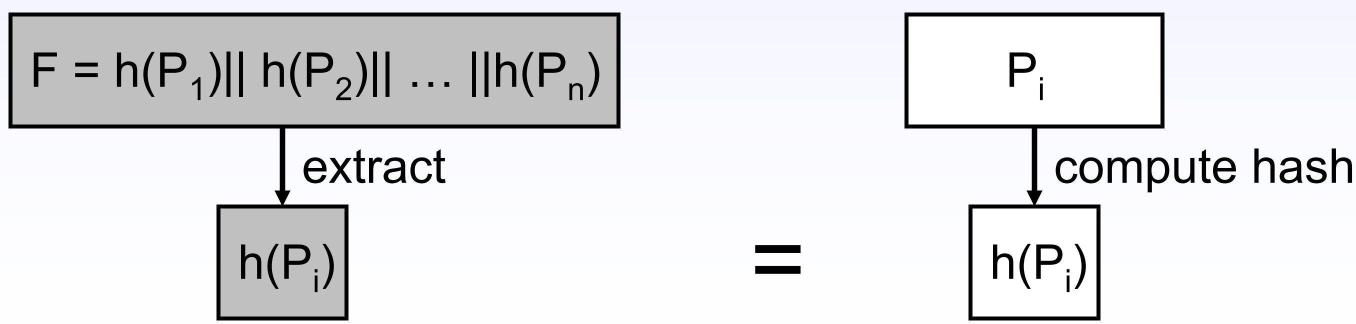


Signature Amortization using IDA

Verification



For each packet P_i, verify:



Delays

- Sender
  - Must append information to *n* packets before sending
- Receiver
  - Must receive *m* packets to authenticate and use
  - (Frequently, all *m* packets should arrive approximately at the same time)
- Consequences
  - Approximate additional delay of the time span of each block
  - For minimal delay, we need smaller block size

Practical Costs - Computation

	Computational costs per block		**Operations possible per second**
	Sender	Receiver	Pentium 2.4 GHz
Erasure encodes	1	0	2,755
Erasure decodes	0	1	3,700
RSA-1024 signature generations	1	0	25
RSA-1024 signature verifications	0	1	1,170

We can send approximately one block every 40 ms.

Acceptable Delay

The International Telecommunications Union – Telecommunications Standardization Sector states the following maximum end to end transmission times that they consider “allowable” with echo control. (Recommendation G.114) [5]

Delay	**Acceptability.**
0 - 150 ms	acceptable to most user application.
150 - 400 ms	acceptable when the impact on quality is aware of.
400 ms	unacceptable

Practical Costs - Bandwidth

Given:

- n/m = 1.5
- using RSA-1024
- 20 byte SHA-1 hash
- blocks of 64 packets (unencoded) of size 1024 bytes (65536 bytes total)

Bandwidth overhead =

2112 bytes per block

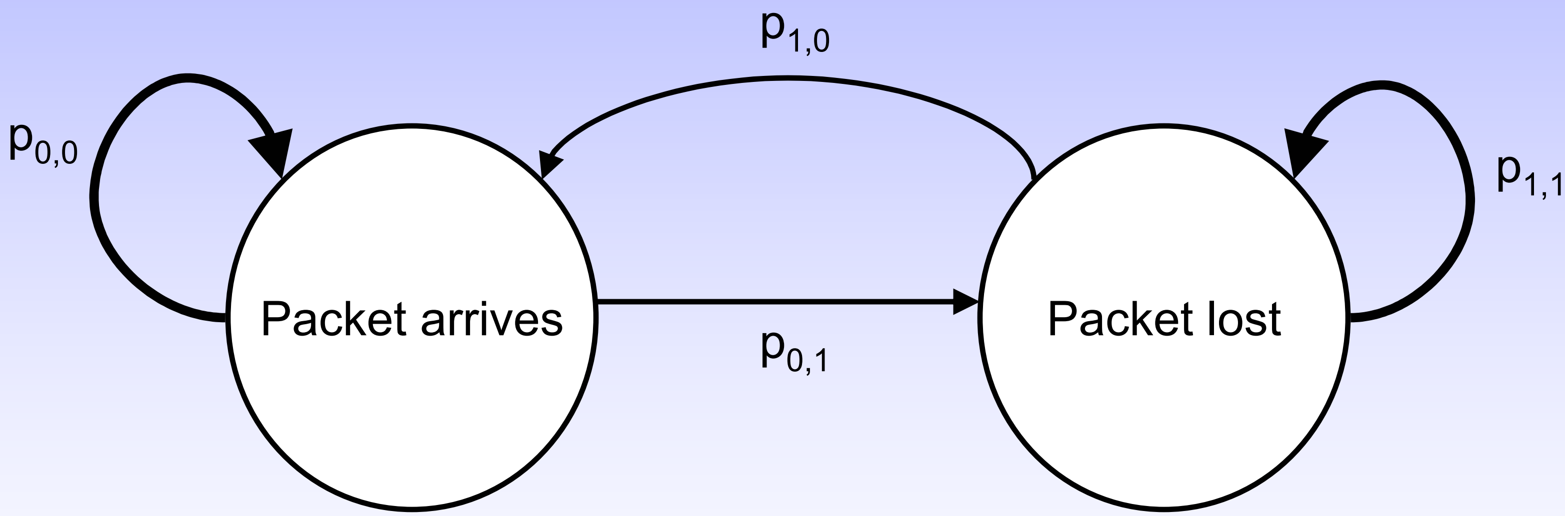
3.2%

Conclusion: Costs are extremely reasonable in the simple case.

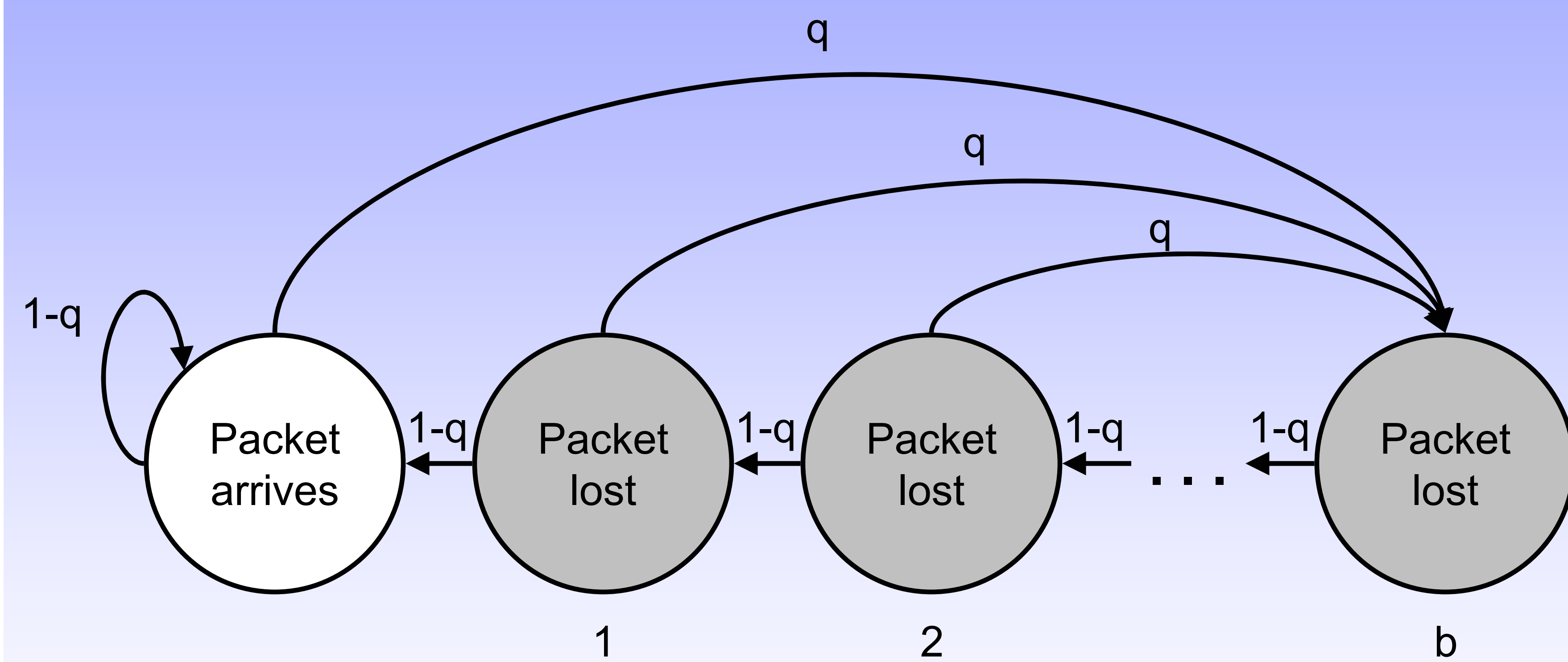
Authentication Probability

- Burst losses are an important part of their analysis
- 2 models
  - 2 state Markov chain model (2-MC)
  - “Biased coin toss”

2 State Markov Chain Model (2-MC)



Biased Coin Toss Model



Internet Traffic Loss

“probe” packets sent from University of Massachusetts, Amherst [17]

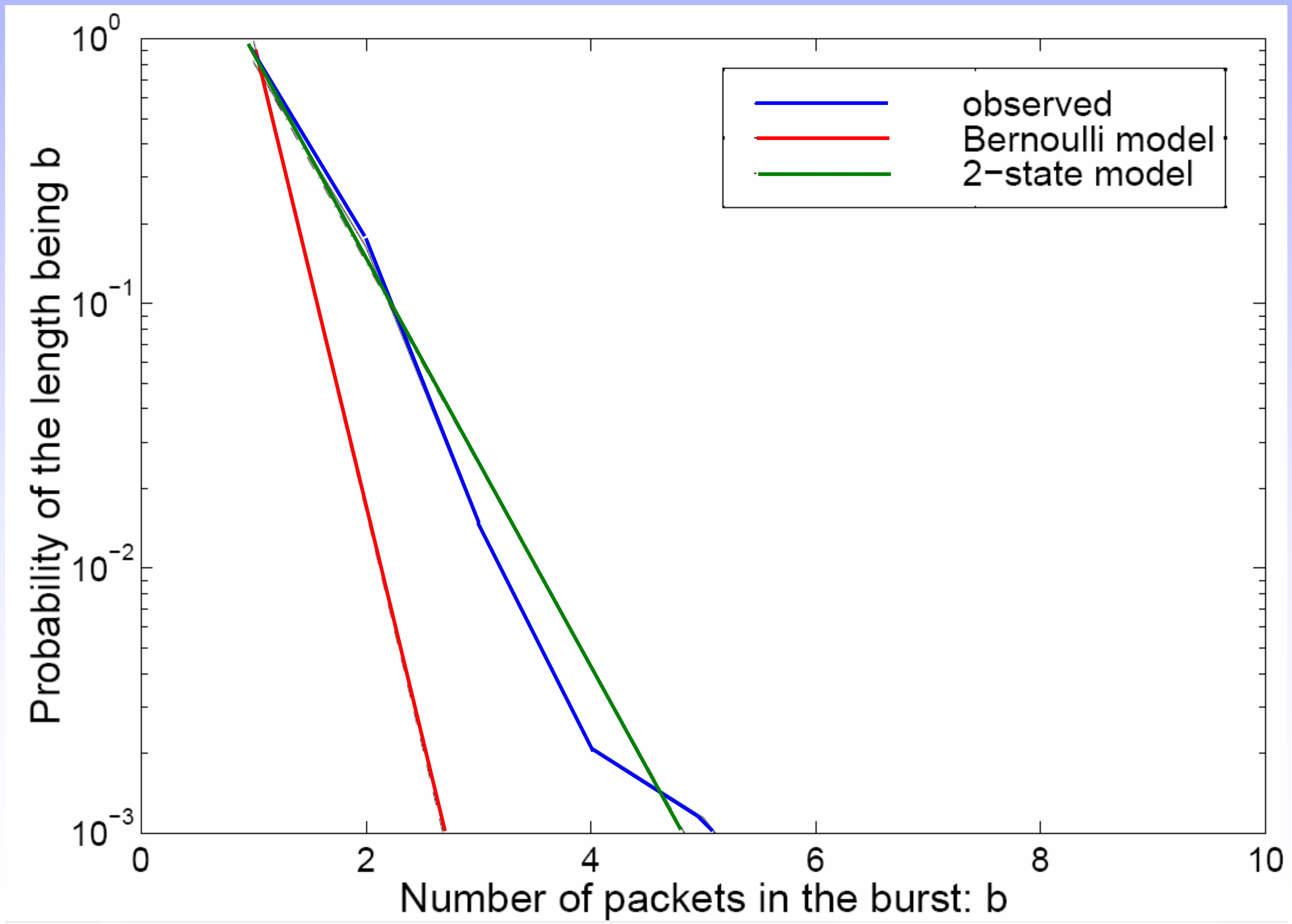
number of destinations in multicast is unknown…

time given in eastern daylight time

date	time	type	destination	duration	sending interval	loss %
Nov ’97	09:52	unicast	SICS, Sweeden	8 hrs	80 ms	2.7%
Nov ’97	09:53	multicast	SICS, Sweeden	8 hrs	80 ms	11.0%
Dec ’97	13:41	unicast	Seattle	2.5 hrs	20 ms	1.7%
Dec ’97	13:39	multicast	Seattle	2.5 hrs	20 ms	3.8%

Internet Traffic Loss

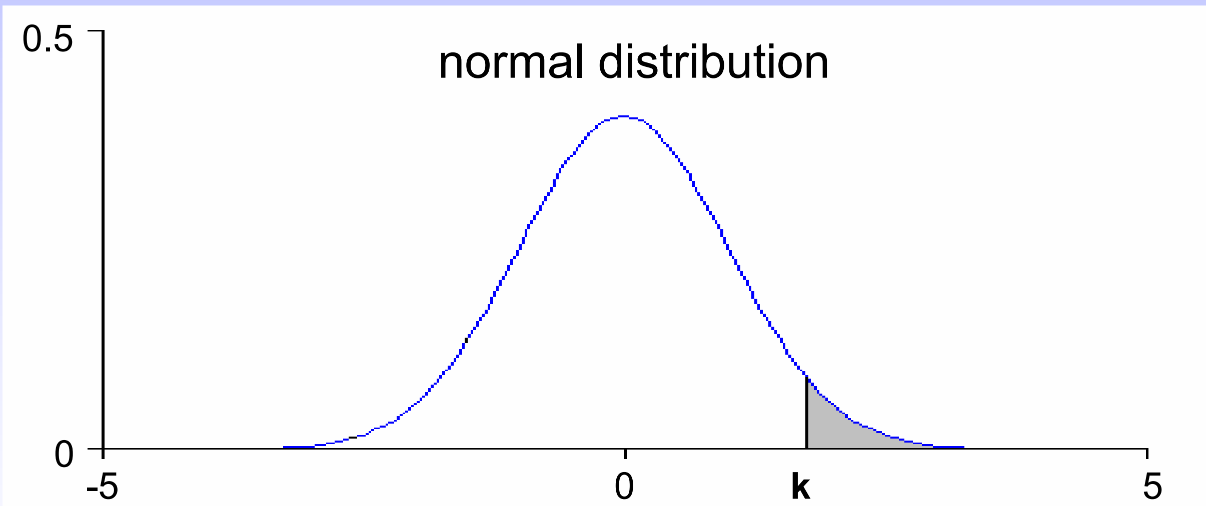
Distribution of packet loss bursts for the Seattle unicast data



Authentication Probability Results

Authentication probability $\rightarrow \Phi(k)$

as $n \rightarrow \infty$



$$k = \frac{n}{m}\sqrt{n}\ (mumble)$$

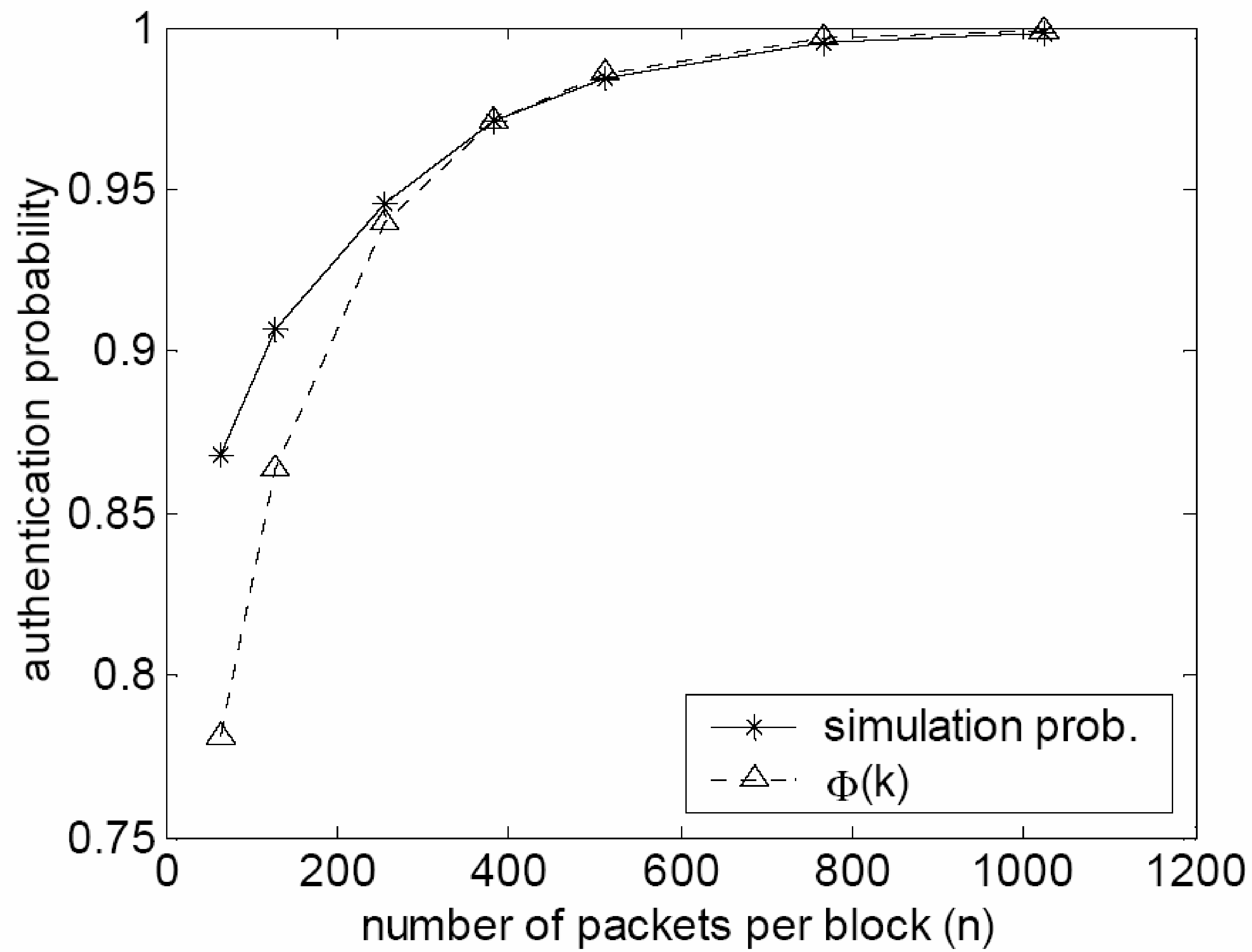
Authentication Probability vs. Block Size

2-MC model

$$\frac{n}{m} = 1.5$$

20% packet loss rate

Expected burst length of 8

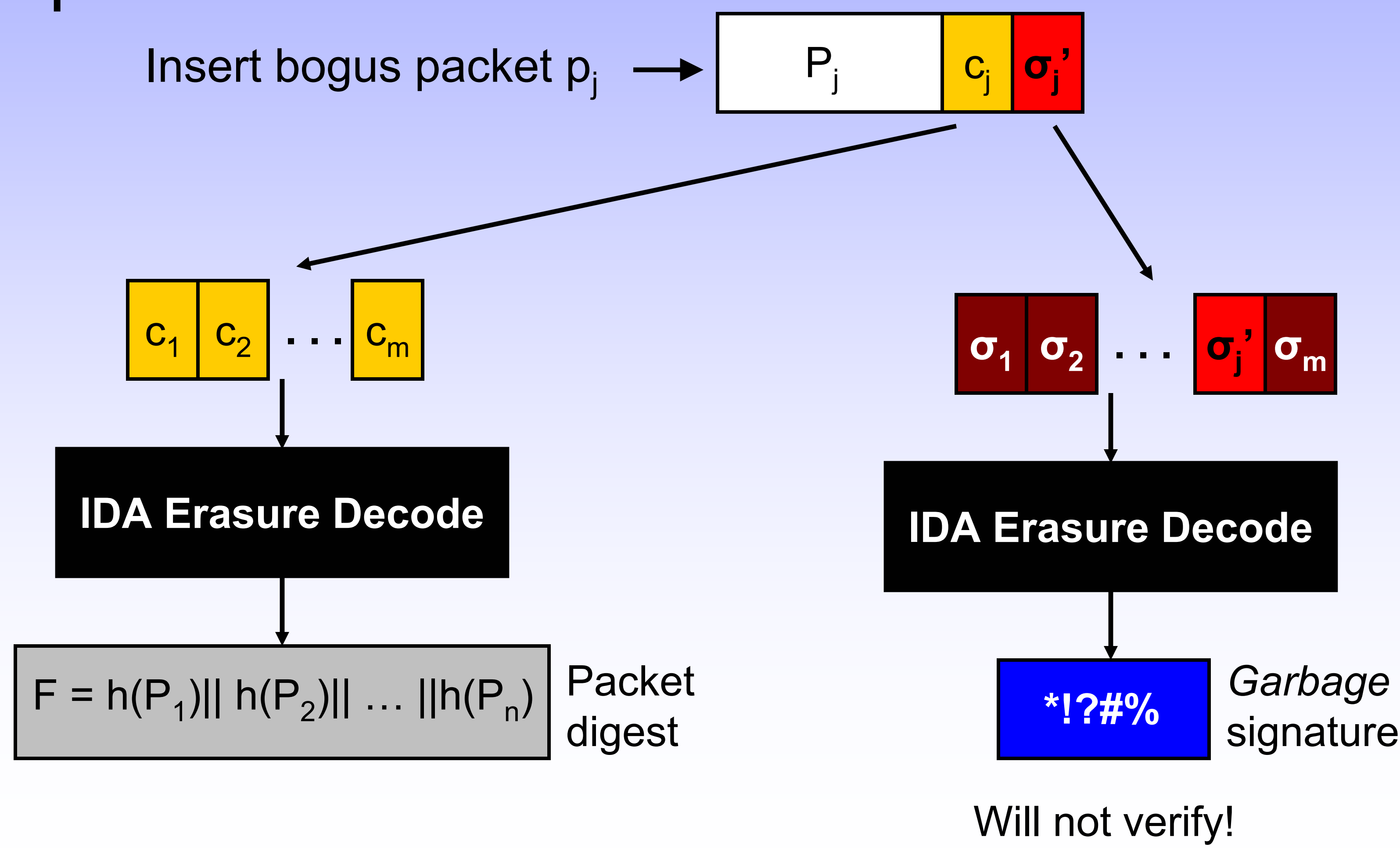


Problem – DoS on SAIDA

A single bogus packet will prevent the authentication of an entire block.

Problem – DoS on SAIDA

Simple attack

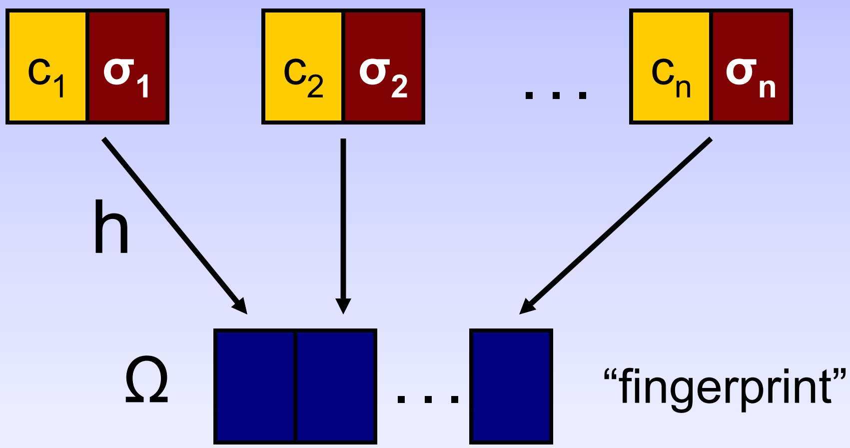


Possible Solution – Error Correcting Codes (ECCs)

- Similar to erasure codes
  - Encode *m* objects to *n* (*n*>*m*)
  - Receiver decodes back to original message
- Allow for a certain number of errors
- More expensive
  - Computation
  - Size
- Common example: Reed-Solomon [15]
  - Plots a polynomial of degree *m-1* in a field
  - Over-plots the polynomial with redundant points (sends them)
  - Can interpolate through a number of bad points

Addition of Error Correcting Codes

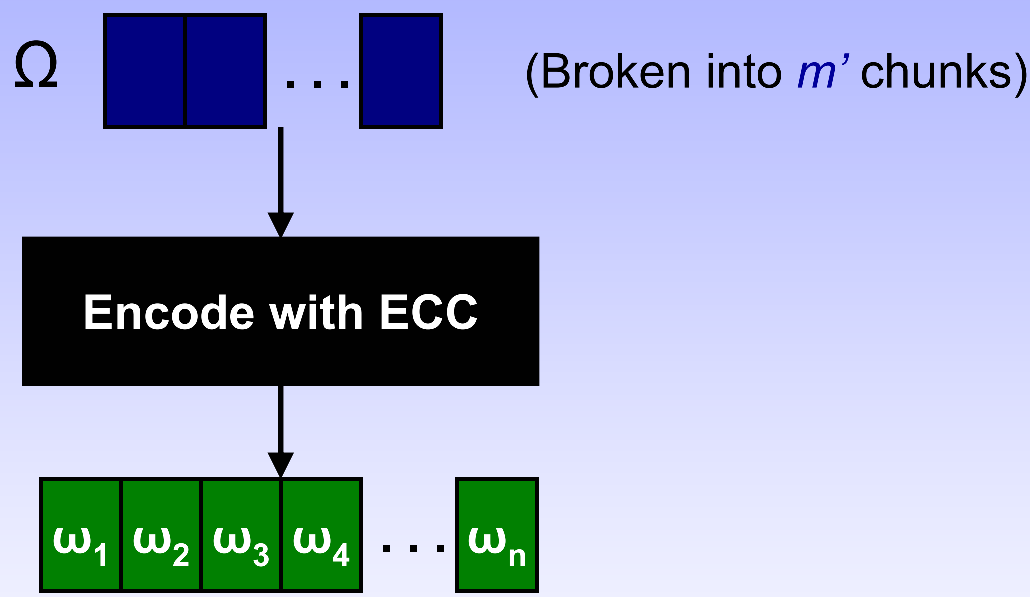
For each block



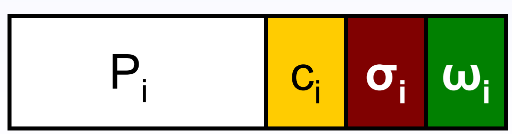
$$\Omega = (h(c_1||\sigma_1)||\ h(c_2||\sigma_2)||\ .\ .\ .\ ||h(c_n||\sigma_n))$$

Addition of Error Correcting Codes

Encode with ECC

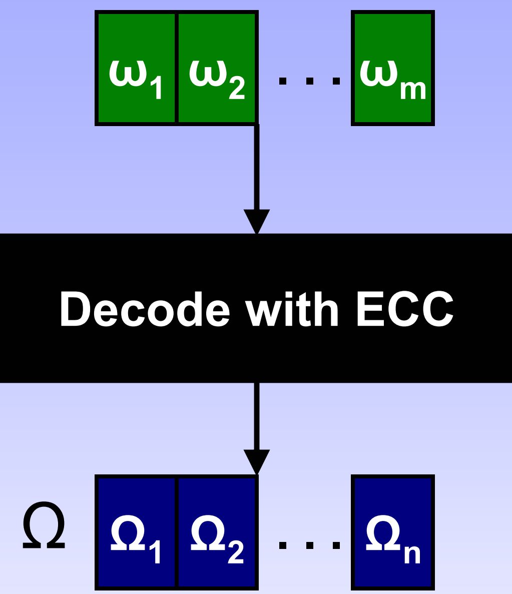


Append to each packet



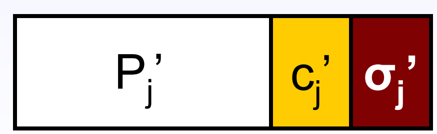
Addition of Error Correcting Codes

Verification



$\Omega = (h(c_1||\sigma_1)||\ h(c_2||\sigma_2)||\ .\ .\ .\ ||h(c_n||\sigma_n))$

For each candidate packet:



verify $\Omega_j = h(c_j'||\sigma_j')$

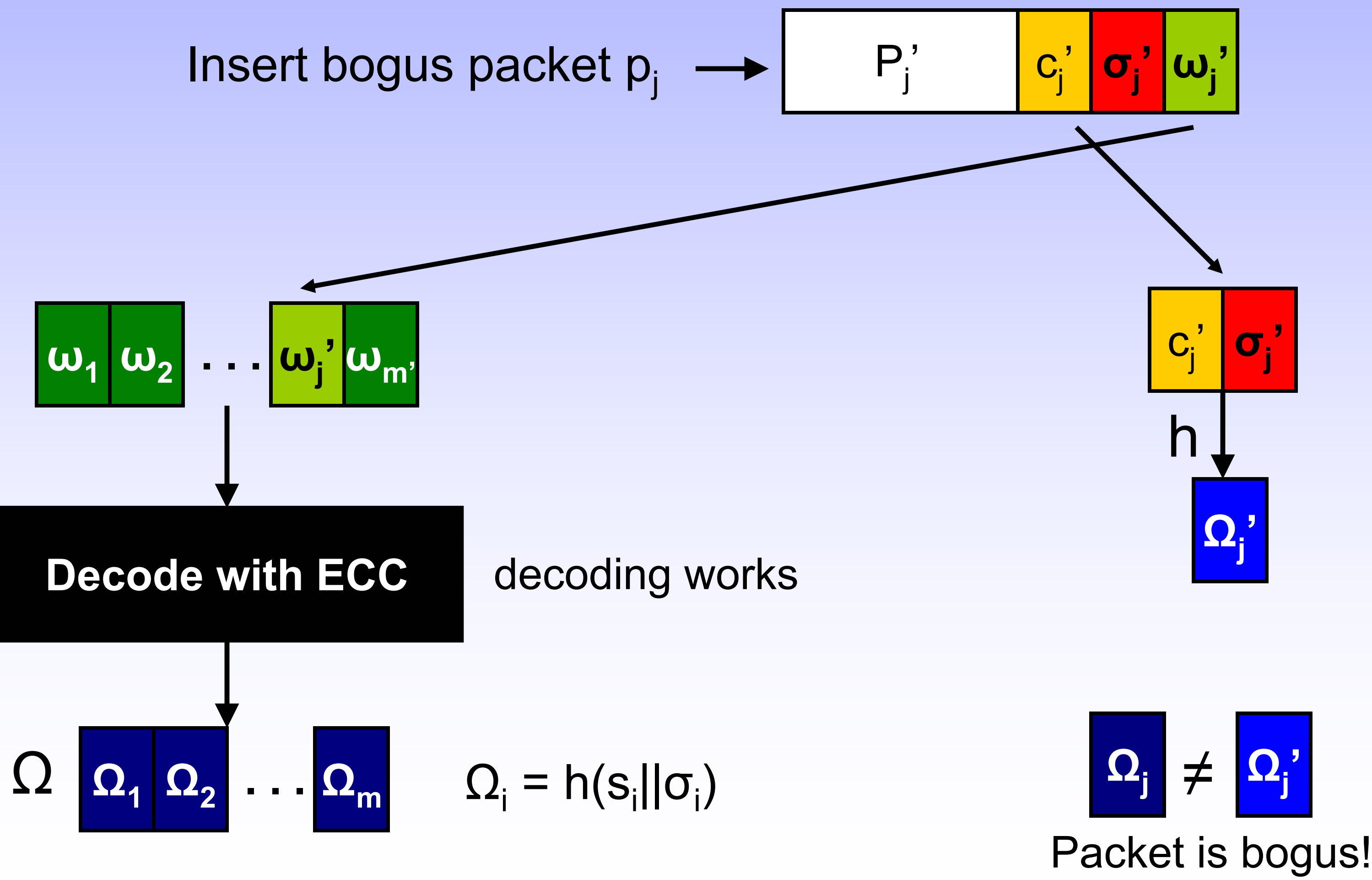
DoS on SAIDA

Claim –

The addition of error correcting codes can prevent the attack where an adversary injects a “small” number of bogus packets into the stream.

DoS on SAIDA with Error Correcting Codes

Attempt attack again



SAIDA Summary

- Pros
  - Ensures use of legitimate packets only
  - Computationally feasible
  - Low bandwidth overhead
  - Good verification probability in burst loss model
  - Withstand attacks when a small number of garbage packets are injected
- Cons
  - Delay at the sender and receiver

Outline

- Three naive solutions
- Brief summary of related work
- Efficient Multicast Stream Authentication using Erasure Codes
- Distillation codes
- Conclusion

Distillation Codes and Applications to DoS Resistant Multicast Authentication [6]

NDSS 2004

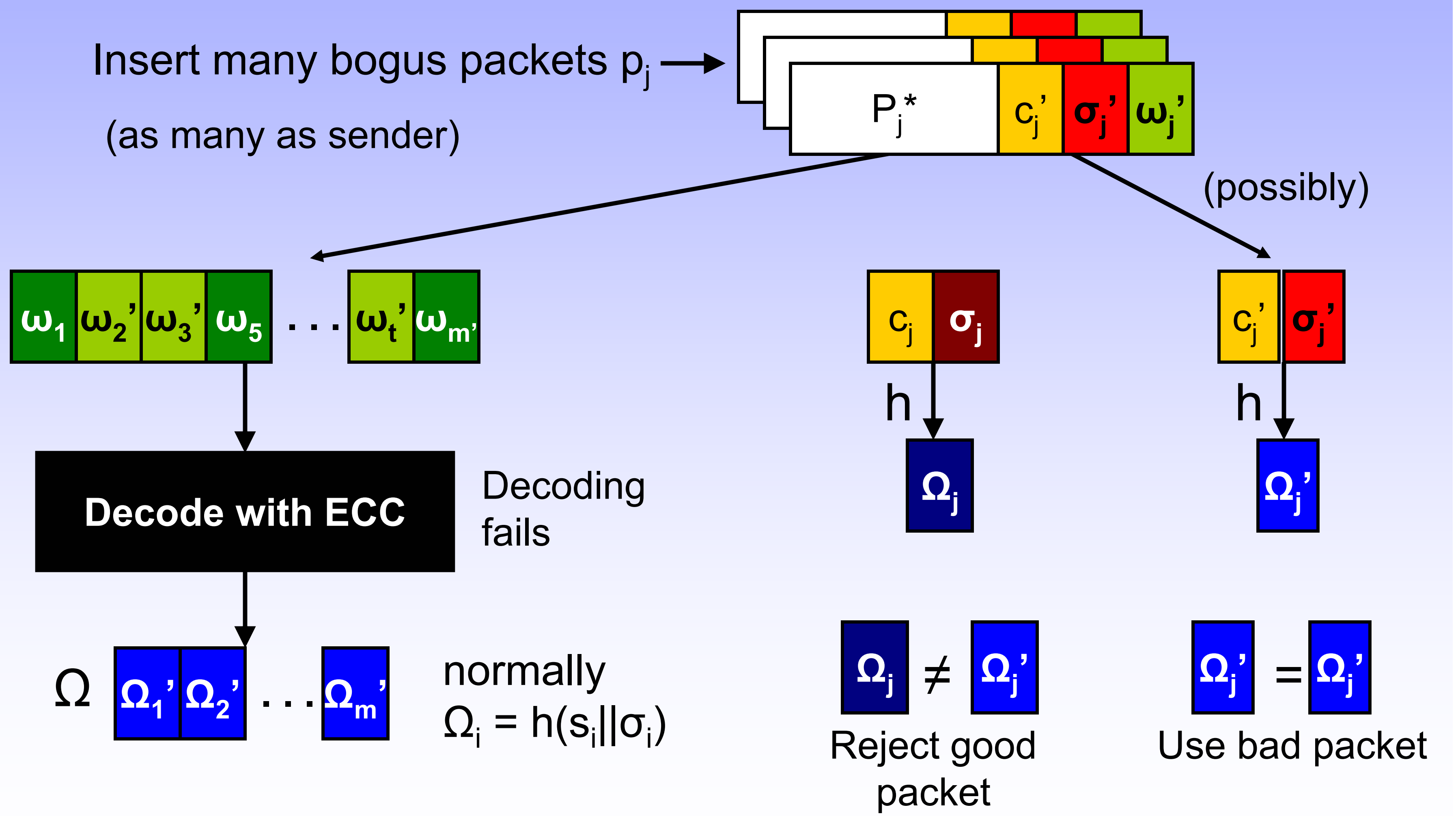
- Objectives
  - **Introduce and address a new adversarial model**
    - **Robustness against pollution attacks (adversary injects many invalid symbols)**
  - “Loss model independent”

Pollution Attack on SAIDA

Claim –

If an adversary injects sufficiently many packets (a number equal to that of the legitimate sender) into a multicast stream, she will launch a complete denial of service attack on the receivers.

Pollution Attack on SAIDA



Strawman Solutions

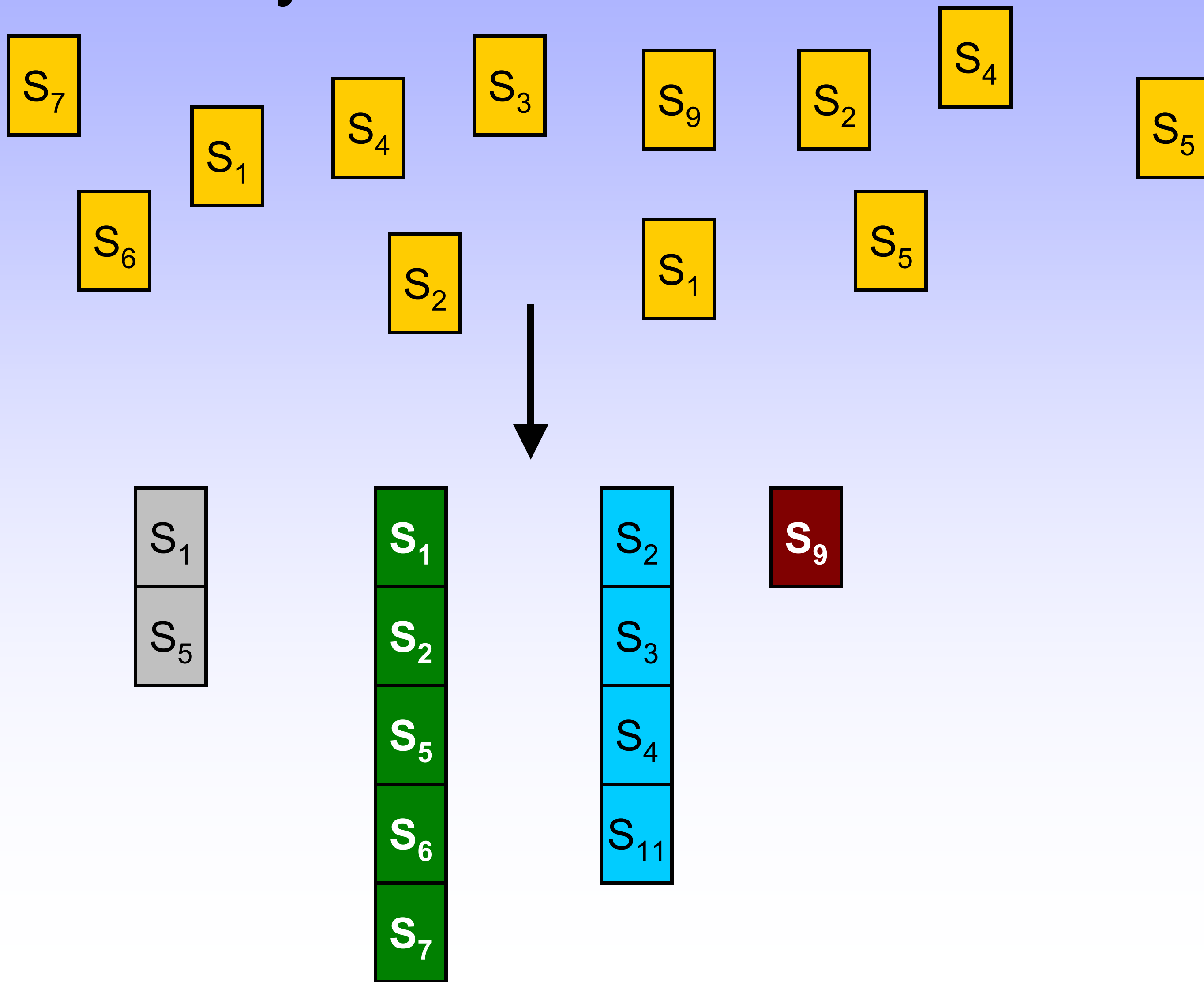
- Decode all possibilities (using erasure code scheme)
  - $\binom{t}{m}$ possibilities (exponential)
- Digitally sign every symbol
  - *Very* expensive (computational, bandwidth)
  - Computational DoS attacks

Distillation Codes

- GOAL: Solve pollution attack vulnerability with "distillation codes"
- General approach
  - Break the packets into groups where all the good packets are exclusively in the same group
  - Compute at most one signature per group
  - Only compute signatures for groups that are sufficiently large
  - This will force a maximum number of signature verifications of only one per *X* packets received.

General Idea

Partition the symbols



Partitioning the Symbols

- Want one partition to contain *exactly* all the valid symbols (*distillation property*)
  - If it is valid, we want it in the right partition
  - Don’t want anyone to be able to create a symbol that could be placed in the same partition as another symbol they simply saw
- Challenges
  - Need secure set membership computed at the receiver’s end

One Way Accumulators

One way accumulator

Assume set S of symbols $\{s_1, \ldots s_n\}$

Accumulate(S) $\rightarrow$ a (accumulator)

Witness(s,S) $\rightarrow$ w

Verify(s,w,a) $\rightarrow$ b (true,false)

Recover(s,w) $\rightarrow$ a

One Way Accumulators

- Must be hard to forge an element of the set

Must be hard to find s', w' where $s' \notin S$ and Recover(s',w') = Accumulate(S)

One Way Accumulator Examples

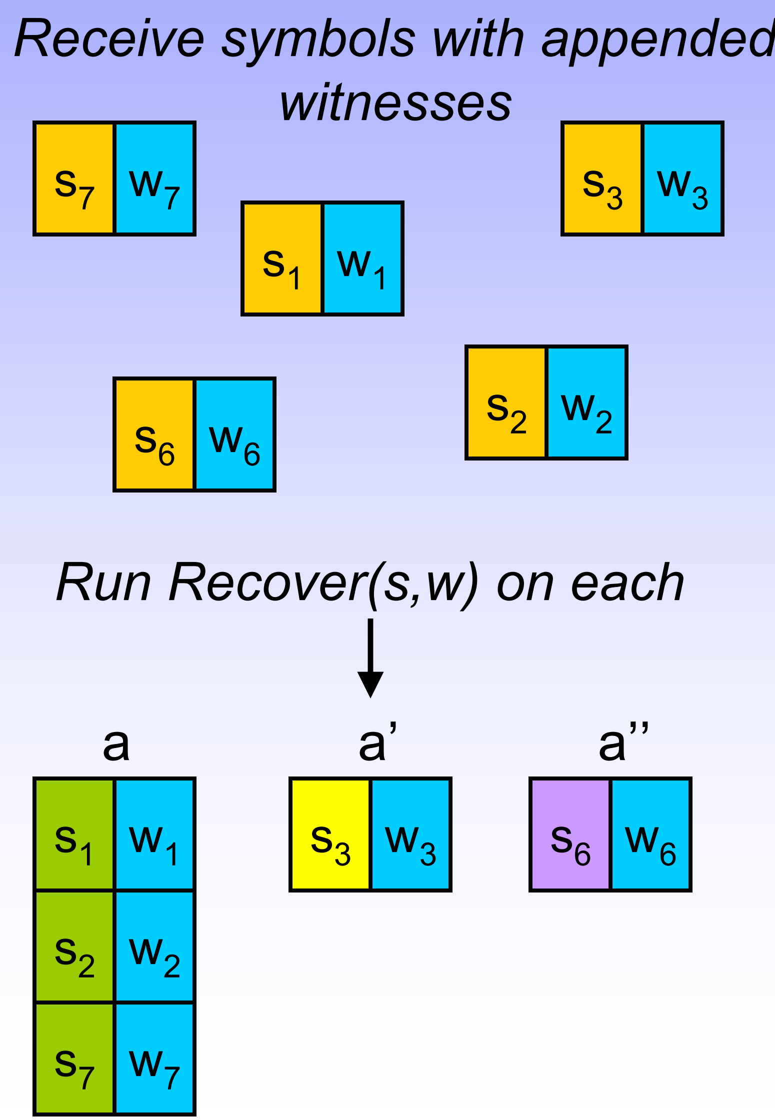
- Merkle hash trees [7]
- Benalod and de Mare *quasi-commutative* one way accumulators [1]
- Camenisch and Lysyanskaya dynamic accumulators [3]

Claim: Given a One Way Accumulator, We Can Partition Symbols

Assume D is our set of valid symbols.

Accumulate(D) → a
Witness(s,D) → w
Verify(s,w,a) → b
Recover(s,w) → a

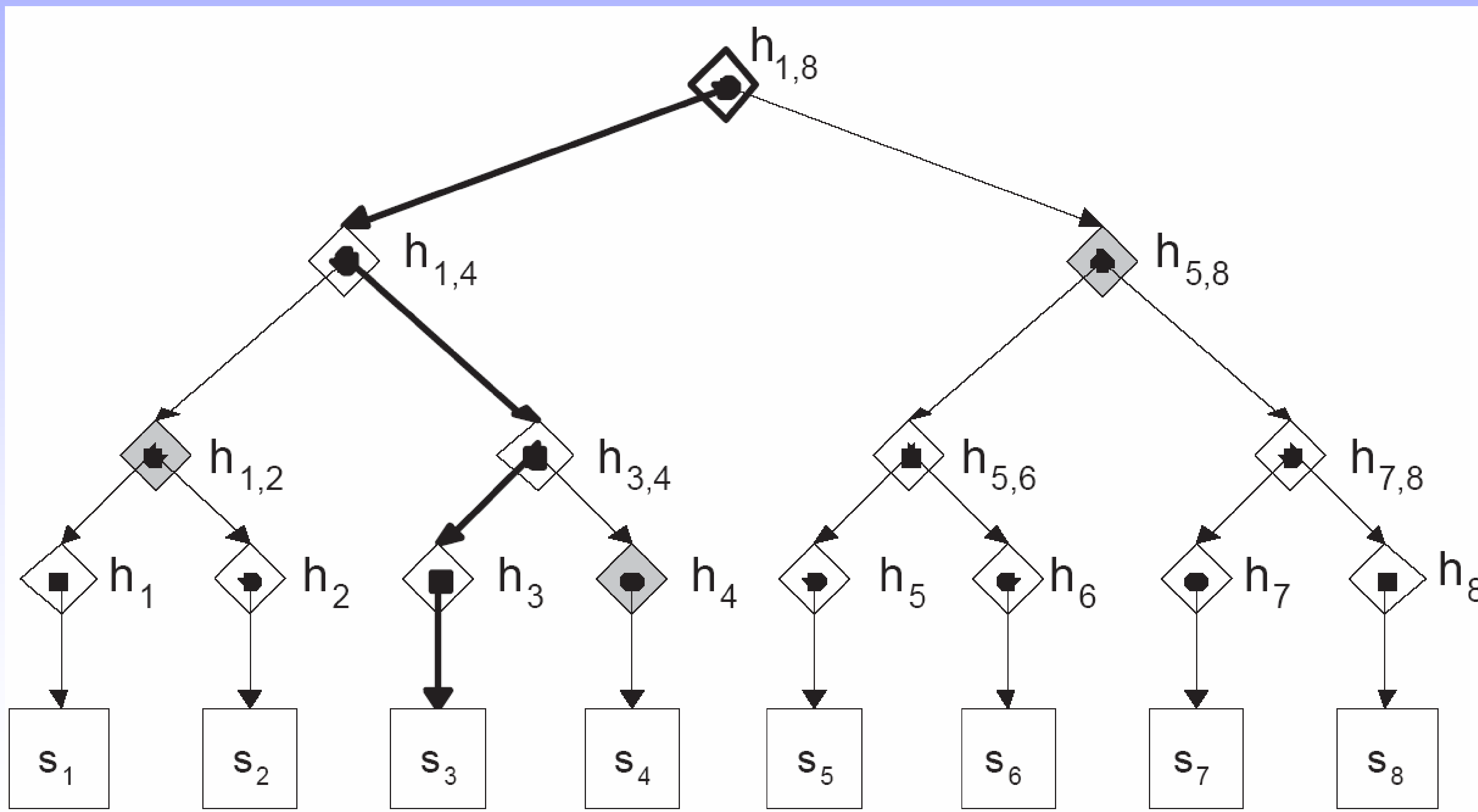
Can't find s' $\notin$ D, and w' where Recover(s',w') = Accumulate(D)



Claim: Given a One Way Accumulator, We Can Partition Symbols

- The definition of Recover(s,w) ensures that all “good” symbols will end up in the same partition.
- The *unforgeability property* of the accumulator ensures that a “bad” symbol cannot be placed in the same partition as the good ones.
- Therefore, with a one way accumulator, we obtain the *distillation property* (by placing each symbol in the partition indexed by its accumulator).

Merkle Hash Trees – Our One Way Accumulator



Merkle Hash Trees – Our One Way Accumulator

Merkle hash trees are a valid one way accumulator.

Finding a Solution to Pollution Attacks…

Putting it all together…

Distillation Code – Definition

An (*n*,*t*) distillation code encodes a message D into a set of *n* symbols $\{s_1, s_2, \ldots s_n\}$ and transmits them over a polluted erasure channel ensuring:

Authenticity. It will never give an invalid reconstruction.

Correctness. If given set of symbols T contains at least *n-t* (*m*) valid symbols of D, then an execution of the decoder on T will output a valid reconstruction.

Distillation Codes - Encoding

Sign the message D

$D' = $ [D | $\text{sig}_{K\_priv}(D)$]

Break D' into m symbols

[d_1 | d_2 | d_3 | d_4] . . . [d_m]

Erasure encode the m symbols

[s_1 | s_2 | s_3 | s_4 | s_5 | s_6] . . . [s_n]

Append witness information

(make single packets)

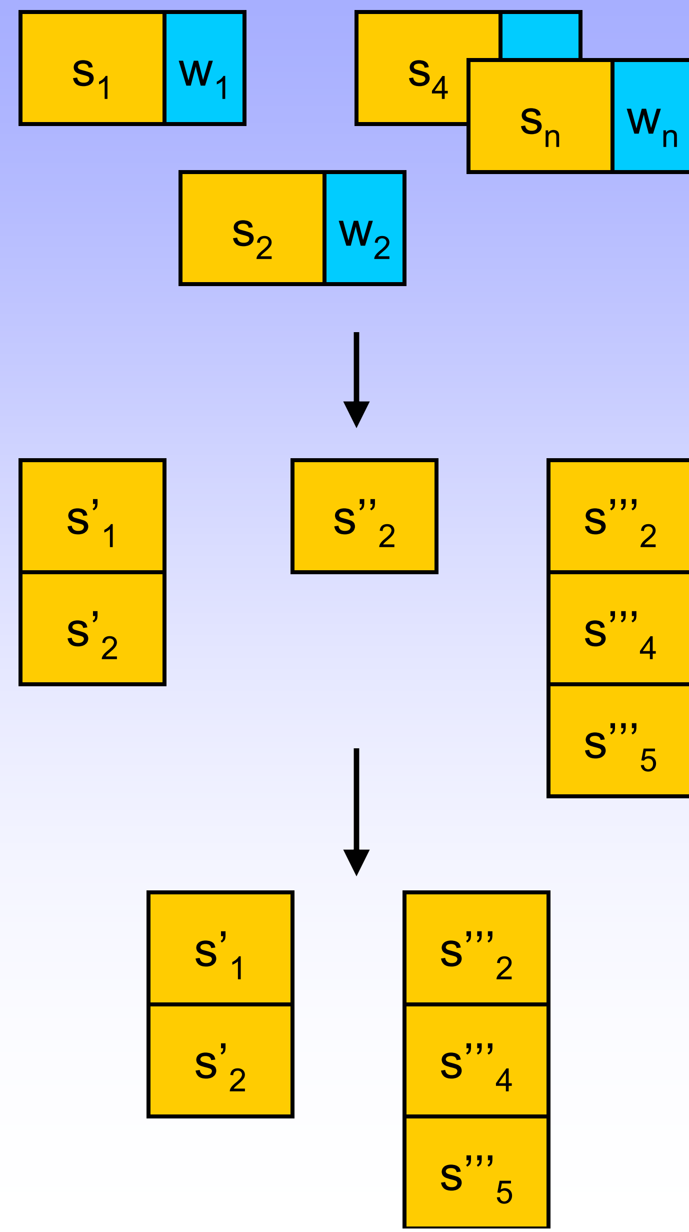
[s_1 | w_1] [s_2 | w_2] . . . [s_n | w_n]

Distillation Codes - Decoding

Partition the received symbols (packets)

Remove witness information

Throw away partitions with less than *m* (say 2)

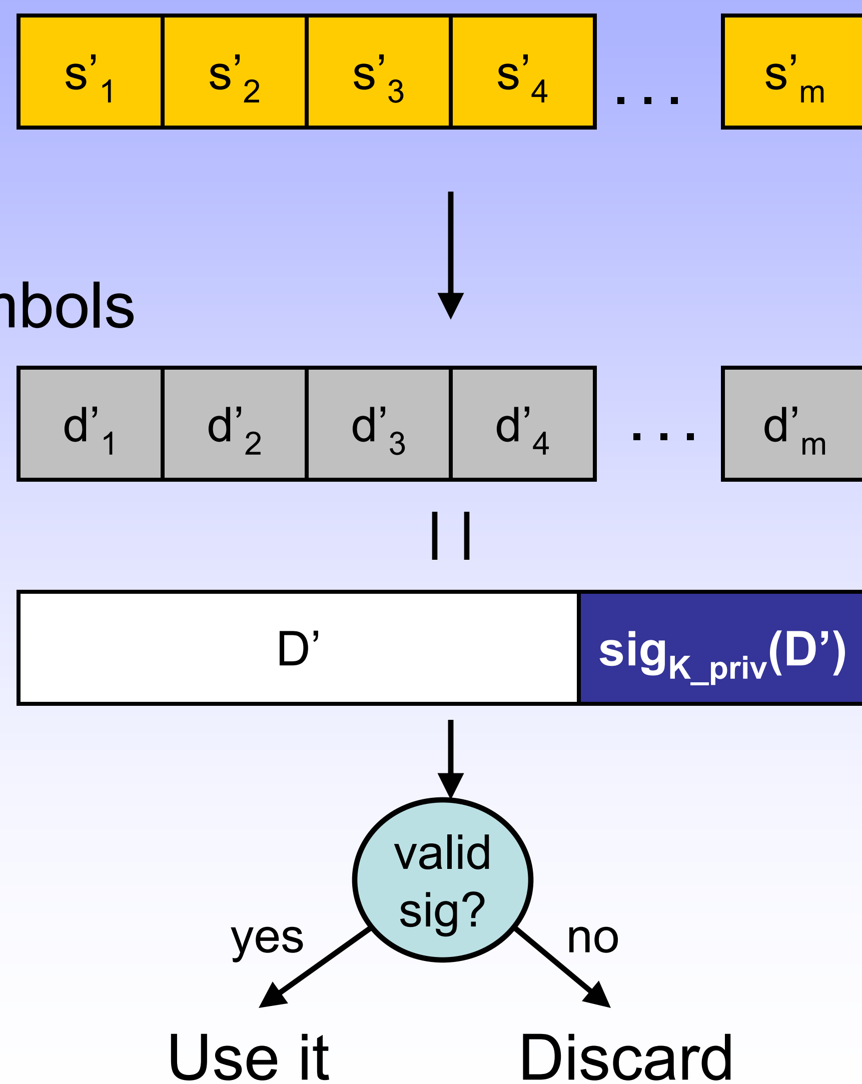


Distillation Codes – Decoding (cont')

For each remaining partition:

Erasure decode the m symbols

Verify whether D' has a valid signature



Distillation Codes

Our construction now satisfies the authenticity and correctness properties. (It is a valid *distillation code*.)

Authenticity. The last step of the decoding is a signature verification. We assume a false signature cannot be generated. (Note that the signature also gives us non-repudiation here.)

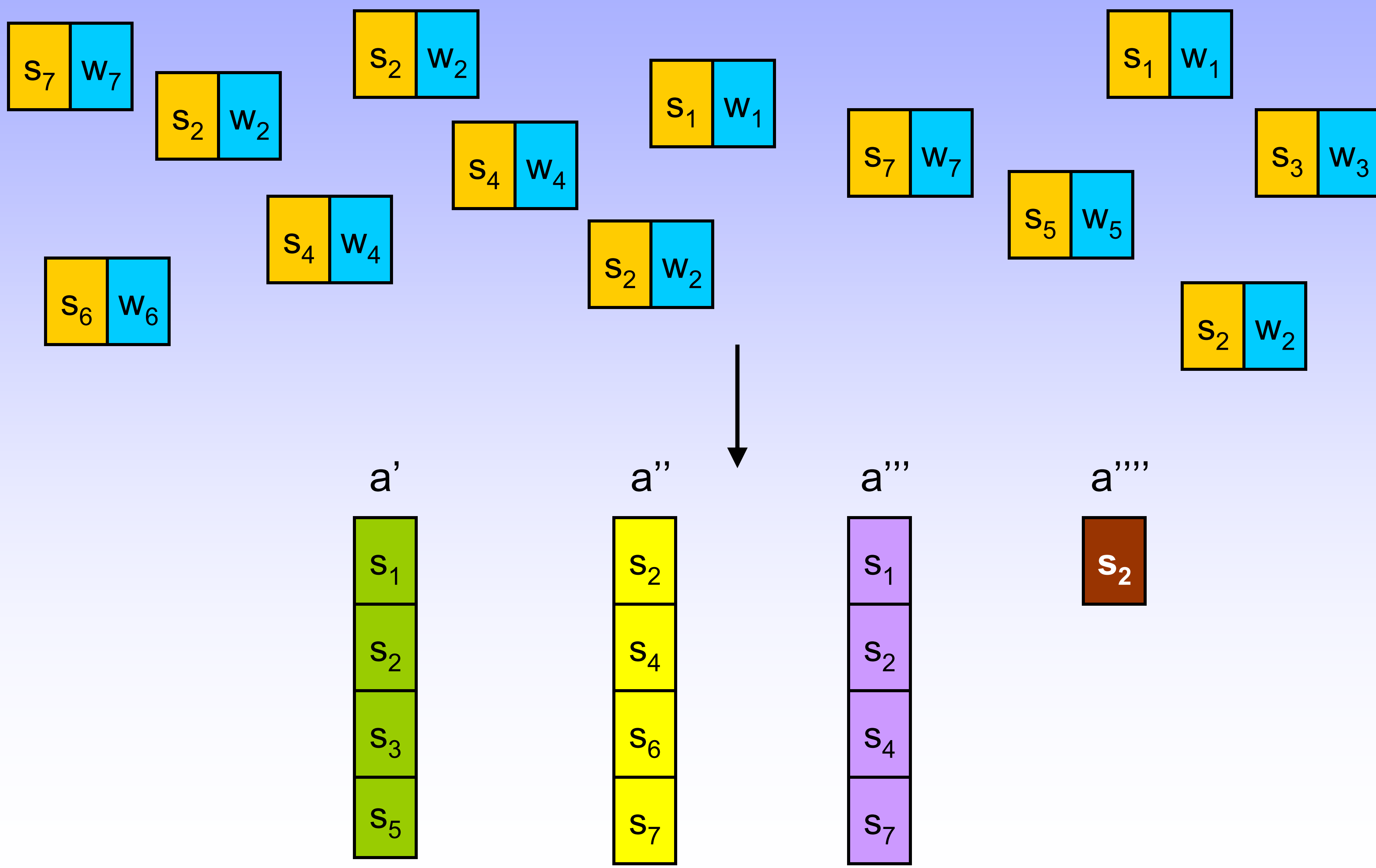
Correctness. We know our partitioning scheme satisfies the distillation property. Therefore, there is one partition that contains *exactly* the valid symbols. Since we verify each partition, we will verify this one.

Resistance to Pollution Attacks

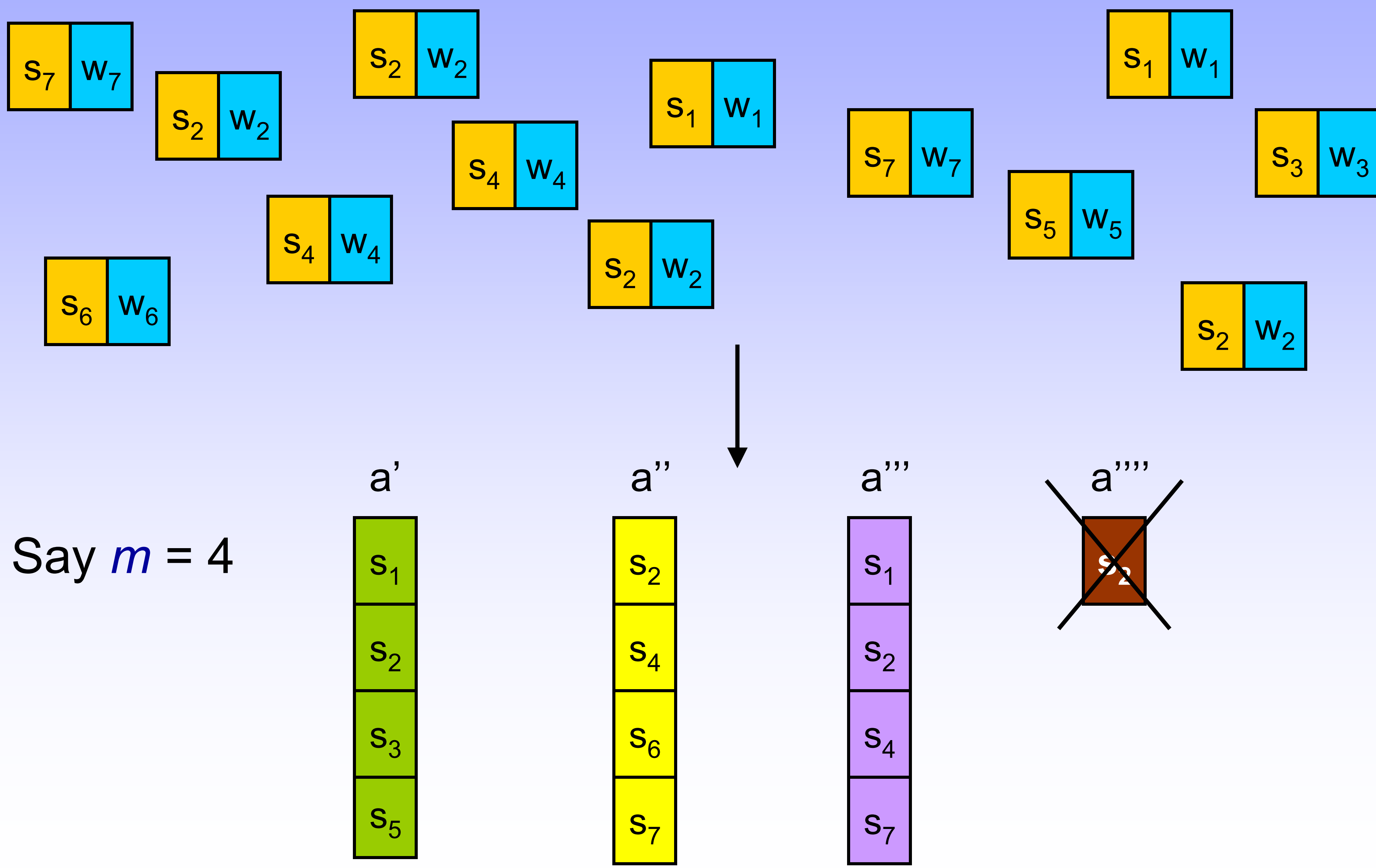
Claim –

A receiver using distillation codes will compute at most one signature for every *m* packets, she receives.

Distillation Codes – Attack Example

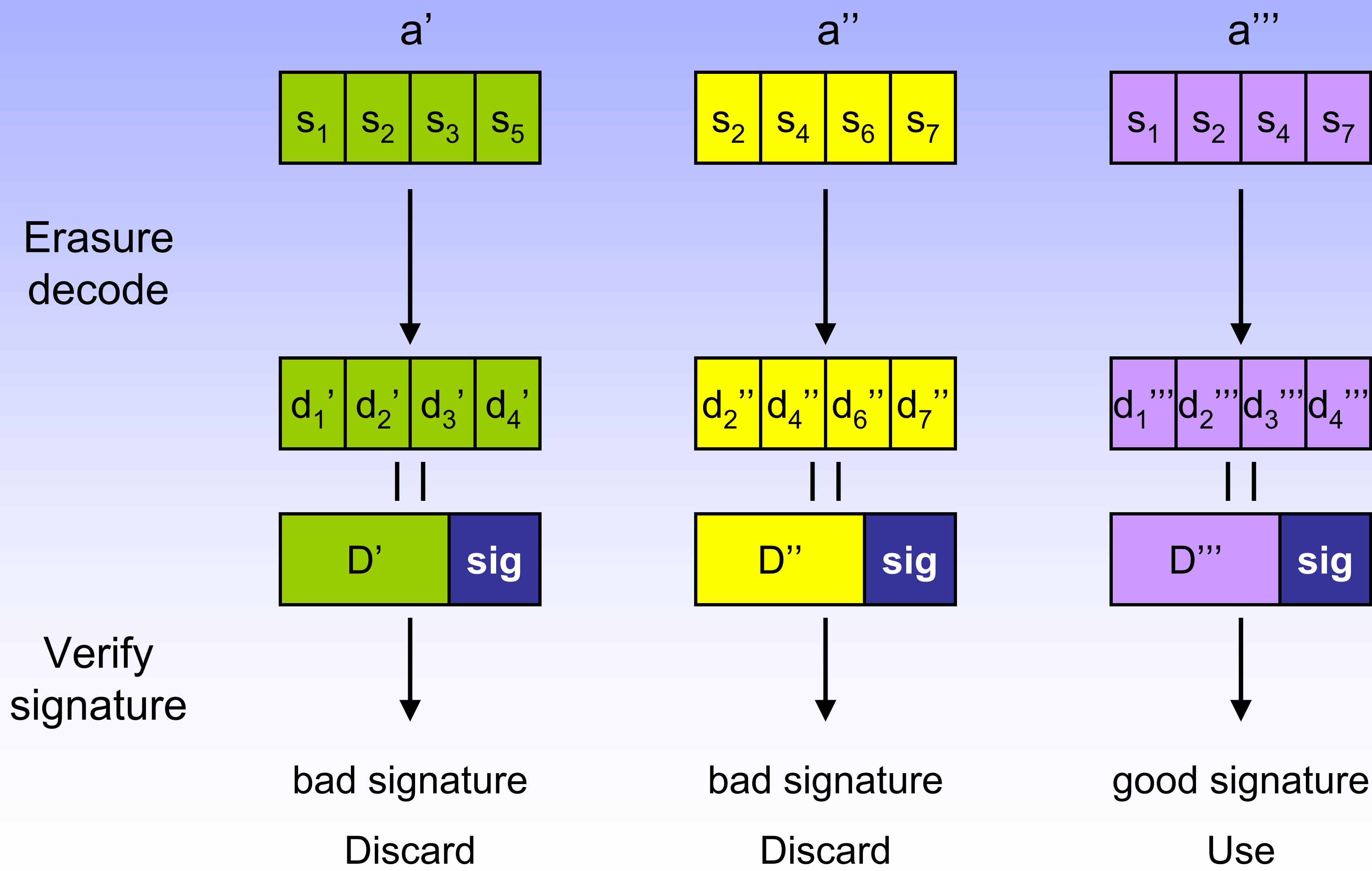


Distillation Codes – Attack Example



Say m = 4

Distillation Codes – Attack Example



Attack Resilience Snapshot

- They claimed attack factor of 10 with 4 Mbs stream required at most 13% CPU in the worst case.
- Assuming a maximum packet delay of 2 seconds, that same stream requires at most **11.87 MB** of memory.

Distillation Codes Summary

- Pros
  - Robust against pollution attacks (with an attack factor of no more than 10)
  - Ensures authenticity (with non-repudiation)
  - Robust to packet loss
  - Small overhead
- Cons
  - (Loss model dependent)
  - Delay at both the sender and receiver
  - Still vulnerable to attacks with more packets

Conclusion

- Information assurance through multicast is challenging
- Can be done at various costs
- Still no perfect solution
- They fit applications well
- Many tools out there that can be useful in various situations

References

[1] J. Benaloh and M. de Mare. One-way accumulators: a decentralized alternative to digital signatures. *Advances in Cryptology (EUROCRYPT ’93).* LNCS, vol. 765, Springer-Verlag, pp.274-285. 1993.

[2] D. Boneh, G. Durfee, and M. Franklin. Lower bounds for multicast message authentication. In *Advancess in Cryptology (EUROCRYPT ’01).* B. Pfitzmann Ed. Springer-Verlag, pp. 437-452. 2001.

[3] J. Camenisch and A. Sysyanskaya. Dynamic accumulators and applications to efficient revocation of anonymous credentials. *Advances in Cryptology (CRYPTO ’02).* LNCS, vol. 2442, Springer-Verlag, pp. 61-76. 2001.

[4] R. Gennaro and P. Rohatgi. How to sign digital streams. In *Advances in Cryptology (CRYPTO ’97).* B.S. Kaliski Jr., Ed. Springer-Verlag, pp. 180-197. 1997.

[5] International Telecommunications Union. Telecommunication Standardization Sector. Recommendation G.114. `http://www.itu.int/ITU-T/`.

[6] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J.D. Tygar. Distillation codes and applications to DoS resistant multicast authentication. *Network and Distributed System Security Symposium (NDSS ’03).* Internet Society. 2003.

[7] R. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Research in Security and Privacy*. pp. 232-246. 1980.

[8] S. Miner and J. Staddon. Graph-based authentication of digital streams. In *IEEE Symposium on Research in Security and Privacy*. pp. 232-246. 2001.

[9] A. Pannetrat and R. Molva. Efficient multicast packet authentication. *In Network and Distributed System Security Symposium (NDSS ’03).* Internet Society. 2003.

References

[10] J.M. Park, E. Chong, and H.J. Siegel. Efficient multicast packet authentication using erasure codes. ACM Transactions on Information and System Security (TIS-SEC). 6(2):258-285. 2003.

[11] A. Perrig. The biba one-time signature and broadcast authentication protocol. In *Eighth ACM Conference on Computer and Communications Security (CCS-8).* pp. 28-37. 2001.

[12] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient and secure source authentication for multicast. In *Symposium on Network and Distributed Systems Security (NDSS '01).* Internet Society, pp. 35-46. 2001.

[13] A. Perrig, R. Canetti, J.D. Tygar, and D. Song. Efficient authentication and signature of multicast streams over lossy channels. In *IEEE Symposium on Research in Security and Privacy*, pp. 56-73. 2000.

[14] M. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM,* vol. 36, 2, pp. 335-348. 1989.

[15] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*. 8(2):300-304, 1960.

[16] C. Wong and S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking (TON).* Volume 7, IEEE Press, pp. 502-513. 1999.

[17] M. Yajnik, S. Moon, J. Kurose, and Towsley D. Measurement and modeling of the temporal dependence in packet loss. In *IEEE Conference on Computer Communications (INFOCOM '99).* IEEE Press. 1999.