Crypto Tutorial

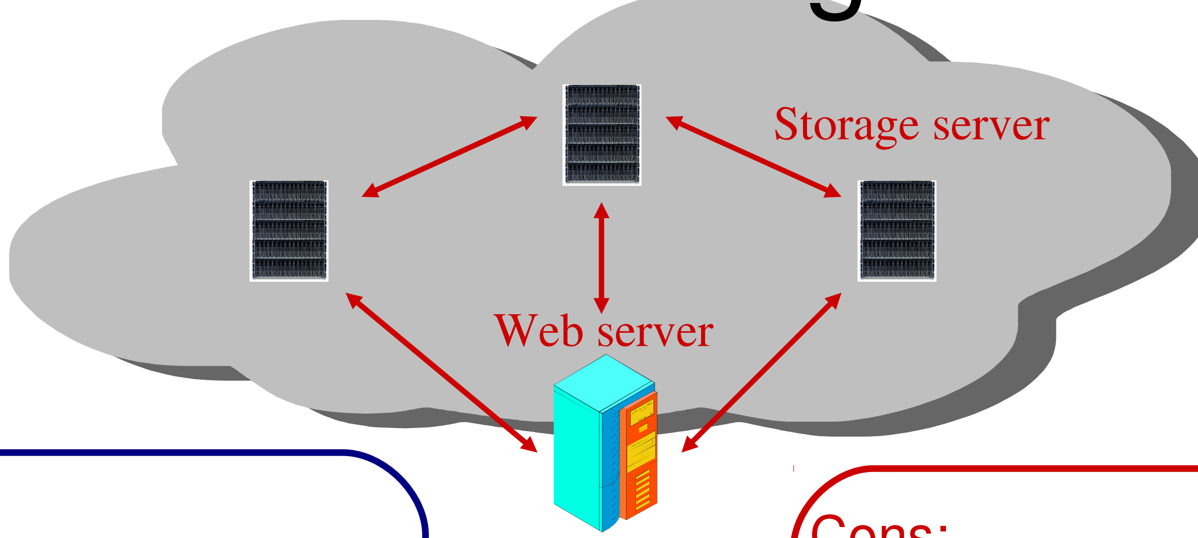
- Homomorphic encryption
- Proofs of retrievability/possession
- Attribute-based encryption
- Hidden-vector encryption, predicate encryption
- Identity-based encryption
- Zero-knowledge proofs, proofs of knowledge
- Short signatures
- Broadcast encryption
- Private information retrieval

Homomorphic encryption (whiteboard)

Proofs of Retrievability

Cloud storage

Cloud Storage Provider

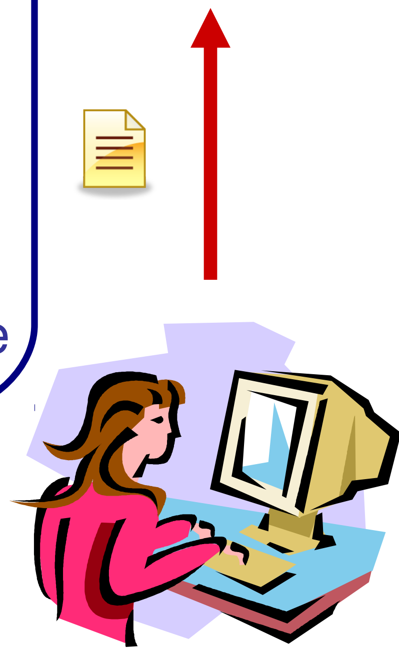


Pros:

- Low cost
- Easier management
- Enables sharing and access from anywhere

Cons:

- Loose direct control
- Not enough guarantees on data availability
- Providers might fail

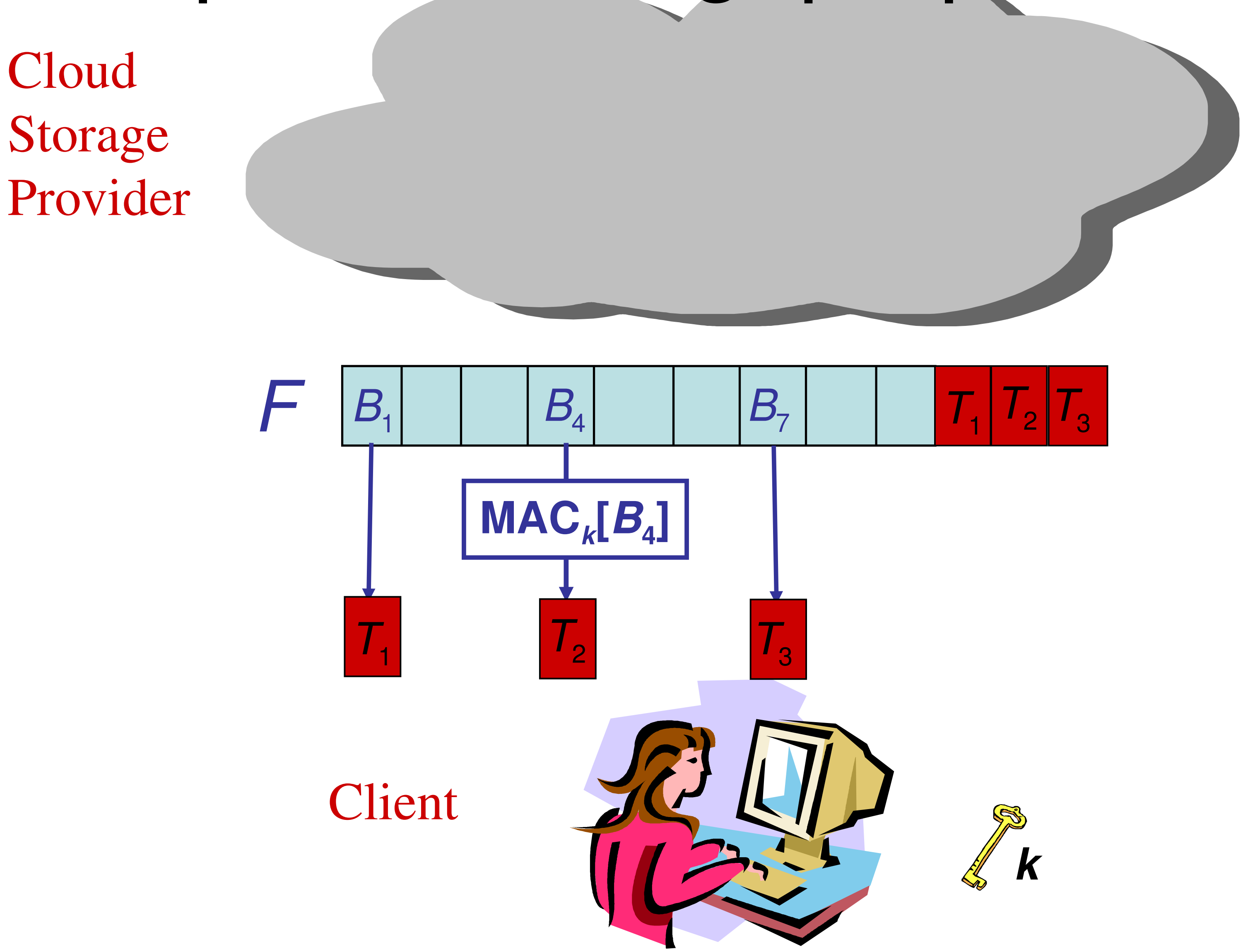


Client

PORs: Proofs of Retrievability

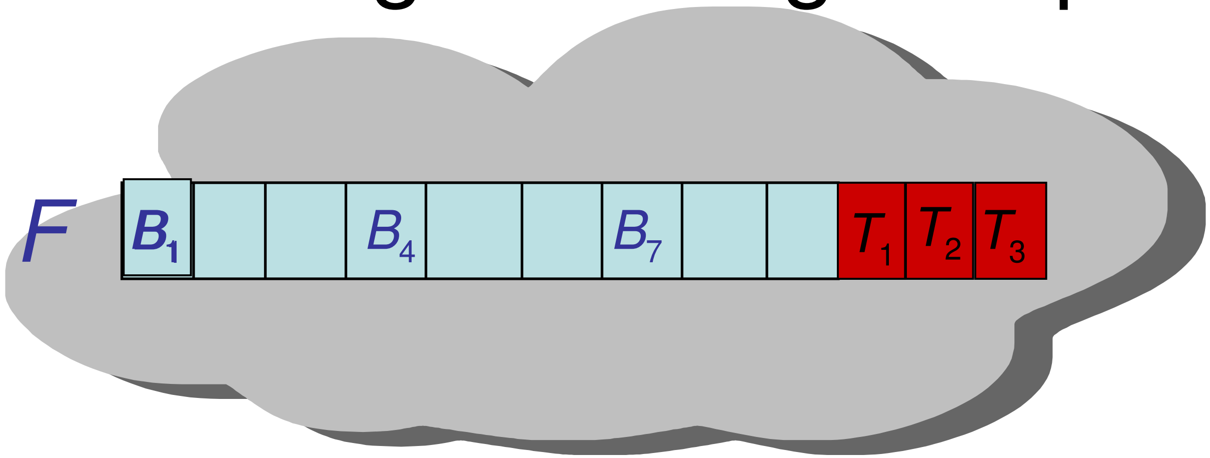
- Client outsources a file *F* to a remote storage provider
- Client would like to ensure that her file *F* is retrievable
- The simple approach: client periodically downloads *F;* This is resource-intensive!
- What about spot-checking instead?
  - Sample a few file blocks periodically
  - If file is not stored locally, need verification mechanism (e.g., MACs for each file block)
  - Probabilistic guarantees

Spot-checking: preparation



Spot-checking: challenge-response

Cloud
Storage
Provider

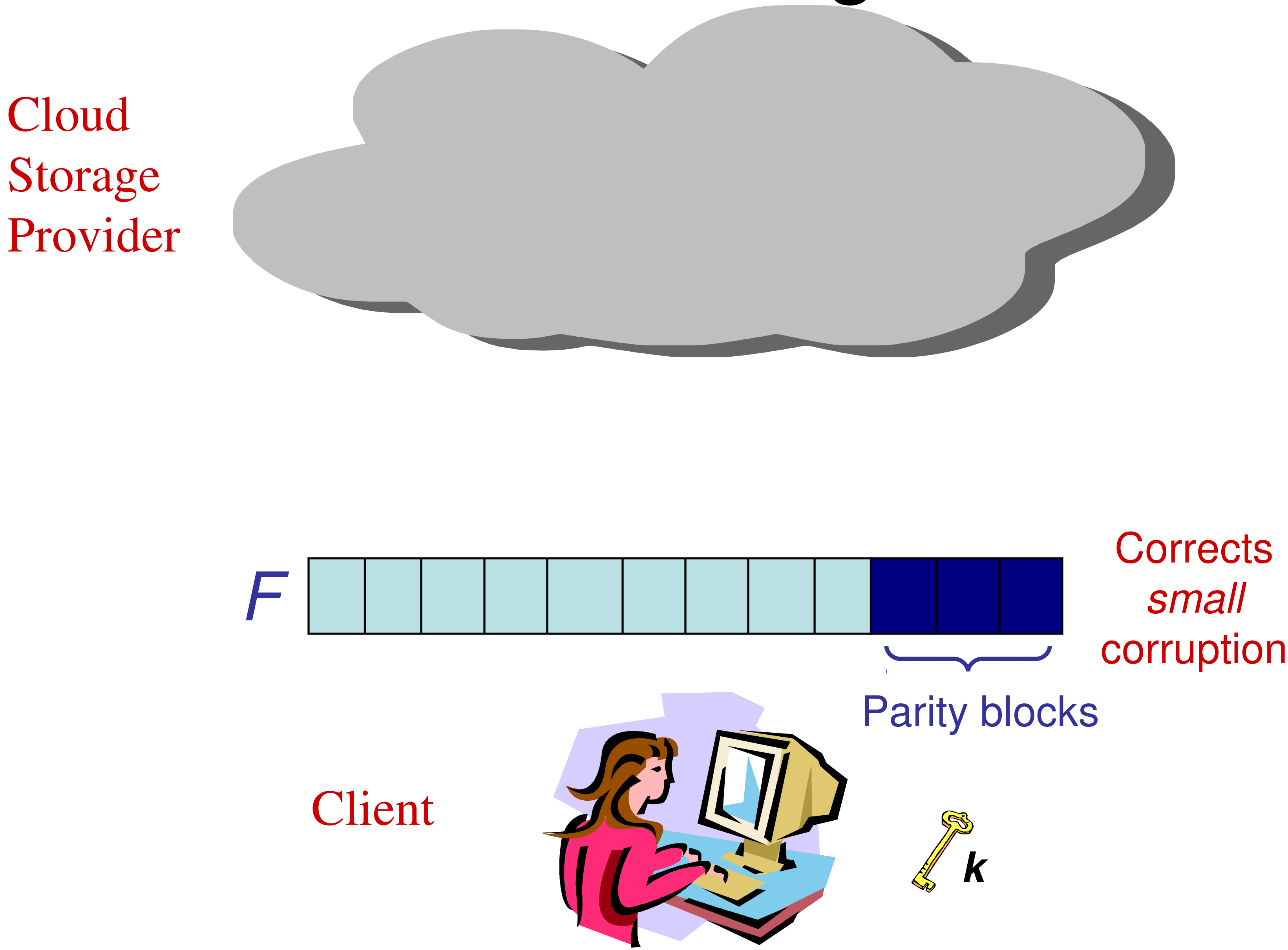


Cons: it does not detect small corruption to the file

Client

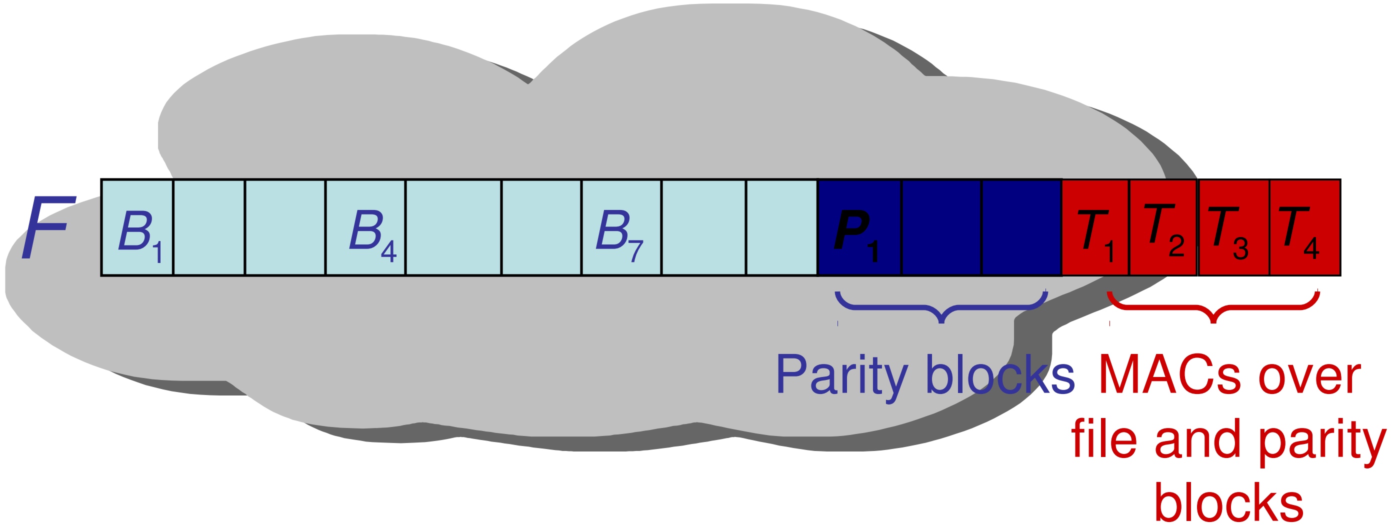
k

Error correcting code



ECC + MAC

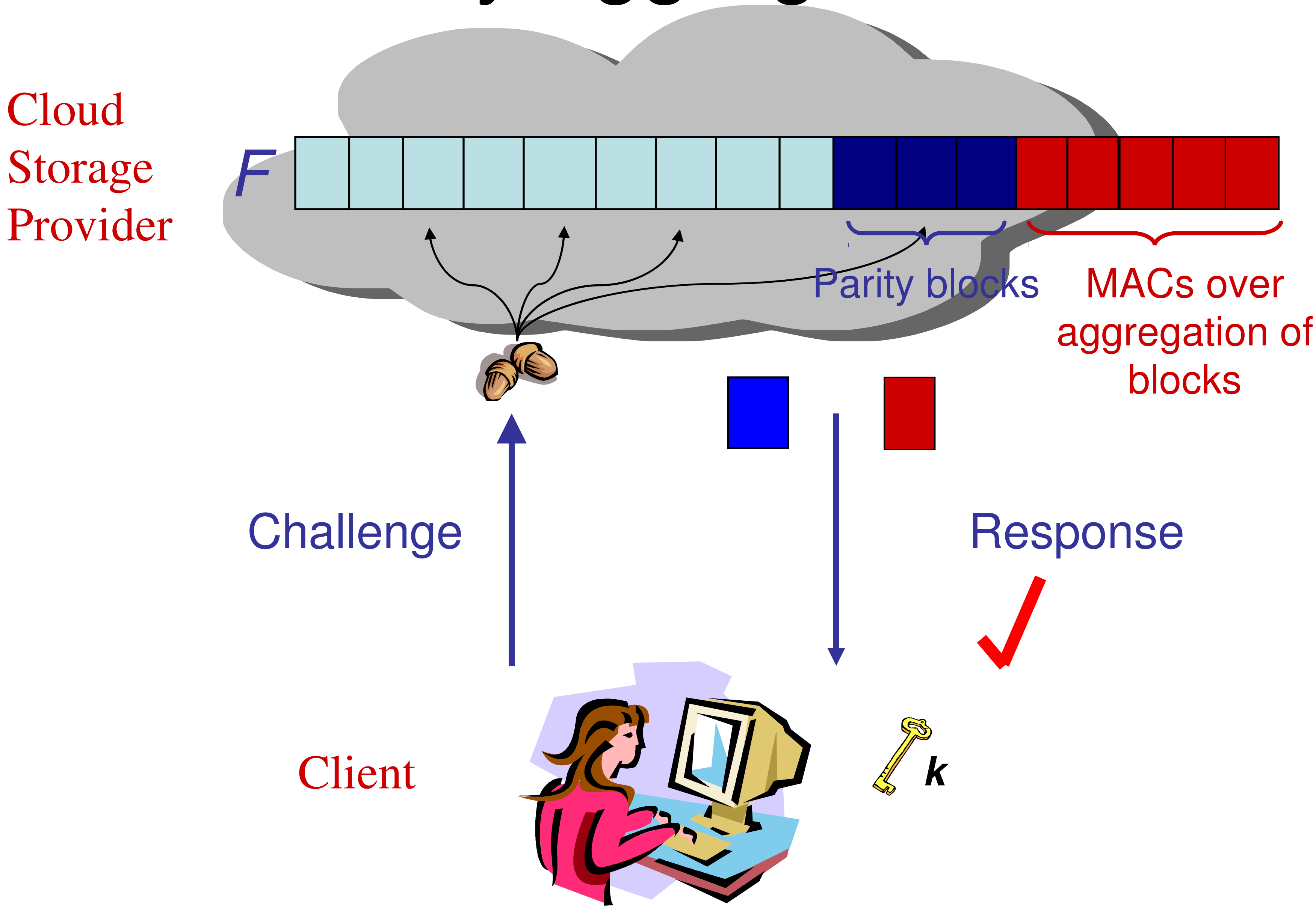
Cloud Storage Provider



- Detect large corruption through spot checking
- Corrects small corruption through ECC

Client

Query aggregation



POR papers

- Proofs of Retrievability (PORs):
  - Juels-Kaliski 2007
  - Enables file recovery for small corruption and detection of large corruption
- Proofs of Data Possession (PDPs)
  - Enables detection of large corruption of file
  - Burns et al. 2007
  - Erway et al. 2009
- Unlimited queries using homomorphic MACs: Shacham-Waters, 2008; Ateniese, Kamara and Katz 2009
- Fully general query aggregation in PORs
  - Bowers, Juels and Oprea 2009; Dodis, Vadhan and Wichs 2009

Practical considerations

- Apply ECC to a large file (e.g., 4GB) is expensive
  - One-time operation
  - Custom built code based on striping and Reed-Solomon
  - Encoding speed of up to 5 MB/sec (could be further optimized)
  - Additional storage overhead due to ECC and pre-computed MACs ≈ 10% (configurable)
- Challenge-response based on spot checking
  - Bandwidth and computationally efficient
  - Challenge and response size on the order of up to 100 bytes
- Example
  - Failure probability 10^{-6}, 4GB file, 32 byte blocks
  - 10% storage overhead
  - Read 100 blocks in a challenge (≈ 3KB)
  - Aggregation: linear combination of 100 blocks of size 32 bytes

Attribute-based Encryption Predicate Encryption (with Hidden-vector Encrytion)

Attribute-Based Encryption

- Example:
  - Encrypted files for untrusted storage
  - User should only be able to access files if she has certain attributes/credentials
  - Don’t want to trust party to mediate access to files or keys

- Introduced by Sahai Waters ‘05

Key-Policy vs. Ciphertext-Policy

- Key-policy:
  - Message encrypted under set of attributes
  - User keys associated with access structure over attributes

- Ciphertext-policy:
  - Message encrypted under access structure
  - User keys associated with set of attributes

Key-Policy ABE

- Algorithms:
  - Setup -> PK, SK
  - Encrypt(PK, M, S) -> CT
  - KeyGen(SK, A) -> TK_A
  - Query(TK_A, CT) -> M if $S \in A$,

    $\perp$ otherwise

- Goyal Pandey Sahai Waters '06, Ostrovsky Sahai Waters '07

Ciphertext-Policy ABE

- Algorithms:
  - Setup -> PK, SK
  - Encrypt(PK, M, A) -> CT
  - KeyGen(SK, S) -> TK_S
  - Query(TK_S, CT) -> M if $S \in A$,

    $\perp$ otherwise

- Bethencourt Sahai Waters '07, Goyal Pandey Sahai Waters '08, Waters '08

Predicate Encryption

- Example:
  - Mail server receives email encrypted under user's PK
  - If email satisfies P, forward to pager
  - If email satisfies P', discard
  - Otherwise, forward to inbox

  - Recipient gives server tokens TK_P, $TK_{P'}$ instead of full secret key SK

Predicate Encryption

- Algorithms:
  - Setup -> PK, SK
  - Encrypt(PK, M, x) -> CT
  - KeyGen(SK, f) -> TK_f
  - Query(TK_f, CT) -> M if f(x) = 1,

    $\perp$ otherwise
- Katz Sahai Waters '08: most expressive PE scheme

Hidden Vector Encryption

- HVE is PE with a specific class of predicates f
- Msgs associated with $(x_1,\ldots x_n)$
- Predicates defined by $(a_1,\ldots,a_n)$ where a_i's can be * ("don't care")
- $f_{(a1,\ldots,an)}$ $(x_1,\ldots,x_n)$ = 1 if $a_i = x_i$ or a_i = * for all i

  0 otherwise

- HVE can be used to construct more sophisticated PE schemes

Predicate Encryption vs. ABE

- Predicate encryption similar to key-policy ABE
- ABE hides message but does NOT hide attributes
- PE hides message AND attributes

Identity-based encryption

Identity-Based Encryption

- Public-key encryption in which an individual's public key *is* their identity
- No need to look up someone's public key!
  - No problems with untrusted keyservers
  - No problems with fake public keys
  - No setup required to communicate with a new person

Identity-Based Encryption

- In a normal public-key system, individuals generate their own public/secret *key pair*
- So in an IBE, if the public keys are fixed by the identity, how does one get the corresponding secret key?
- Trusted third party!

Identity-Based Encryption

- Master setup: T runs *MasterKeyGen()*, gets (PK_M, SK_M), *and publishes* PK_M
- Individual setup: T runs *KeyGen*(SK_M, ID_A), gets SK_A, and gives SK_A to A
- Encryption: *Encrypt*(ID_A, PK_M, m) $= x$
- Decryption: *Decrypt*(x, SK_M) $= m$
- The usual security definitions for public-key encryption apply (given assumptions about T).

Identity-Based Encryption - Variants

- Hierarchical identity-based encryption
  - An individual can act as a trusted third party and distribute keys derived from their own secret
  - End up with a hierarchy—a “tree” of identities
  - An individual can use their key to decrypt any message sent to any ID ultimately derived from their own, i.e. in their “subtree”
- Other identity-based cryptography
  - e.g. signatures

IBE - References

- Boneh, Franklin - *Identity-Based Encryption from the Weil Pairing* (2001)
- Cocks – *An Identity Based Encryption Scheme Based on Quadratic Residues* (2001)
- Gentry, Silverberg – *Hierarchical ID-Based Cryptography* (2002)
- Many others...(Boneh/Boyen 04, CHKP 10, Shamir 84, ...)

Zero-knowledge proofs
Proofs of knowledge

Prelude: Commitment

- Allows Alice to commit to a value x to by giving c(x) to Bob
- Bob does not learn any information from c(x)
- When Alice has to reveal x, she cannot convince Bob that she committed to a different x’

Zero-Knowledge Proofs

- Prover *P* wants to convince verifier *V* that a statement is true...without giving *V* any of his secret information about the statement.
- So *P* and *V* engage in an interactive protocol.
- Basic idea: "cut-and-choose"
- *P* commits to two (or more) values that are a function of his input. *V* selects one, which *P* then reveals.
- The single value doesn't give *V* any information, but might let him catch *P* if he's cheating!

Zero-Knowledge Proofs - Properties

- Informal statement of properties—no math!
- Completeness - “If the statement is true, and all parties are honest, then the verifier accepts.”
- Soundness - “If the statement is false, then no matter what the prover says, the verifier won't accept.”
- Zero-knowledge - “The verifier learns nothing from the interaction with *P*—in particular, he doesn't get any information he couldn't have computed on his own!”

Zero-Knowledge Proofs - Example

- 3-coloring problem: Given a graph consisting of vertices connected by edges, is it possible to color each vertex such that no edge connects two vertices of the same color, using only three different colors?
- Suppose *P* and *V* have a graph, and *P* knows a 3-coloring of that graph.
- *P* wants to convince *V* that the graph is 3-colorable, without revealing any information about the coloring itself.

Zero-Knowledge Proofs - Example

- *P* randomly permutes the colors, and then sends a commitment to each vertex's color to *V*
- *V* picks a single edge
- *P* reveals the (permuted) colors of the endpoints of the edge. *V* checks:
  - The commitment is valid
  - The colors are different
  - The colors are in the valid set of three
  - If these don't hold, or if *P* doesn't follow protocol, *V* rejects

Zero-Knowledge Proofs - Example

- Completeness: If *P* knows a 3-coloring and follows the protocol, *V* will not reject
- Soundness: If *P* doesn't know a 3-coloring, he'll either have to break protocol in some way (which *V* would detect immediately), or hope *V* never picks an edge with two vertices the same color
  - Chance he gets away with it is at most 1-1/|E|
  - Repeat! If you repeat the entire interaction 100|E| times, the chance he can successfully cheat is at most $(1-1/|E|)^{100|E|} \approx e^{-100}$

Zero-Knowledge Proofs - Example

- Zero-knowledge:
  - Since *P* permutes the colors at the beginning of each interaction, the colors revealed during one interaction are independent of the colors revealed during any other interaction
  - At each step, *V* learns two different colors for a pair of adjacent vertices...but due to the color permutation, this is a random pair of colors uncorrelated to anything he's seen before
  - ...so he could have just picked two different random colors for those vertices himself, and gotten a statistically identical view to what *P* shows him!

Zero-Knowledge Proofs - Power

- Why did I pick 3-coloring as the example?
- 3-coloring is *NP-complete*
- So any NP statement can be proven using an interactive zero-knowledge proof!
  - Actually, anything in PSPACE...

Zero-Knowledge Proofs - Efficiency

- You probably don't want to use the NP reduction to 3-coloring in practice.
  - The NP reduction will decrease efficiency, and then you have to run the 3-coloring protocol k|E| times.
- Often it's better to look for a direct zero-knowledge proof of something.
  - Graph isomorphism, etc.

Non-Interactive Zero-Knowledge

- Our protocols required interaction of the prover and the verifier. Can't we have something more akin to a mathematical proof, where the prover writes something down and then any verifier who reads it will be convinced?
- Surprisingly, yes!
- NIZK relies on a "common random string" known to all parties, outside the control of *P*
- If everyone trusts that the CRS is truly random, *P* can write down a NIZK
- In practice, NIZKs tend to be huge.

Proofs of Knowledge

- Remember the 3-coloring example...
- *P* wanted to show that the graph was 3-colorable. But he actually did a bit more than that—*P* showed that not only was the graph 3-colorable, but *he knew a 3-coloring*.
- Related concept to ZK: Proof of knowledge
- *P* can show that he knows some value, without revealing anything about the value itself
- Useful for authentication!

ZK/POK - References

- Goldwasser, Micali, Rackoff – *The Knowledge Complexity of Interactive Proof Systems* (1989)
- Goldreich, Micali, Wigderson – *Proofs That Yield Nothing But Their Validity, or All Languages in NP Have Zero-Knowledge Proof Systems* (1991)
- Ben-Or, et al – *Everything Provable Is Provable in Zero Knowledge* (1988)
- Blum, Feldman, Micali - *Non-Interactive Zero-Knowledge and Its Applications* (1988)
- Schnorr - *Efficient identification and signatures for smart cards* (1989)

Short Signatures

Short Signatures

- Signatures that are short [BLS'01]
  - 160 bits instead of 1024 bits for same security
- Based on elliptic-curve cryptography
- Efficient and simple

e

g, h, G =g^a, H = h^a

g, g^{SK}, e

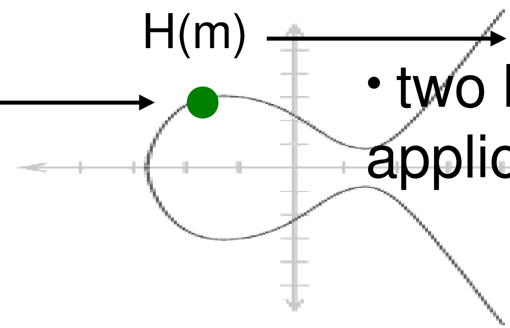
Signer

SK

- a hash computation
- one exponentiation

m

H(m)



Verifier

e

Sig = $H(m)^{SK}$ g^{SK}

- two bilinear map applications

References

- Implementations: C http://crypto.stanford.edu/pbc/
  - Time to sign: 15ms
  - Time to verify: 20ms (but can batch)
  - Comparable to RSA

- References:
  - Short signatures from the Weil pairing – *Boneh et Al., 2001*
  - Pairing-Based Cryptographic Protocols: A Survey, *Dutta et Al., 2004*

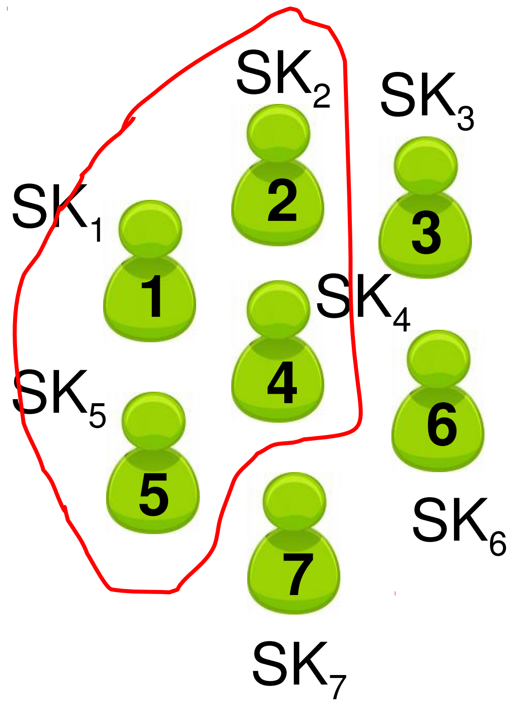
Applications

- Network protocols:
  - Packet size smaller than with RSA
- Integrity of data in outsourced storage

Broadcast encryption

Broadcast encryption

- Encrypting a message such that only a (arbitrary) subset of a group can decrypt it *[Boneh et Al., 2005]*
- Three parts:
  - Setup(no. users) ⟶ secret keys, PK
  - Encrypt(subset, PK) ⟶ (header, K)
    - Send header with encryption
  - Decrypt(header, i, SK_i)
    - Yields K only if i is a member of the subset



Analysis

- *[Boneh et Al, 2005]*: O(√n) ciphertext and public key size
- Implementation in C: http://crypto.stanford.edu/pbc/bce/
- References:
  - J. Horwitz, "A Survey of Broadcast Encryption“, 2003
  - D. Boneh, C. Gentry, and B. Waters, “Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys”, 2005

Applications

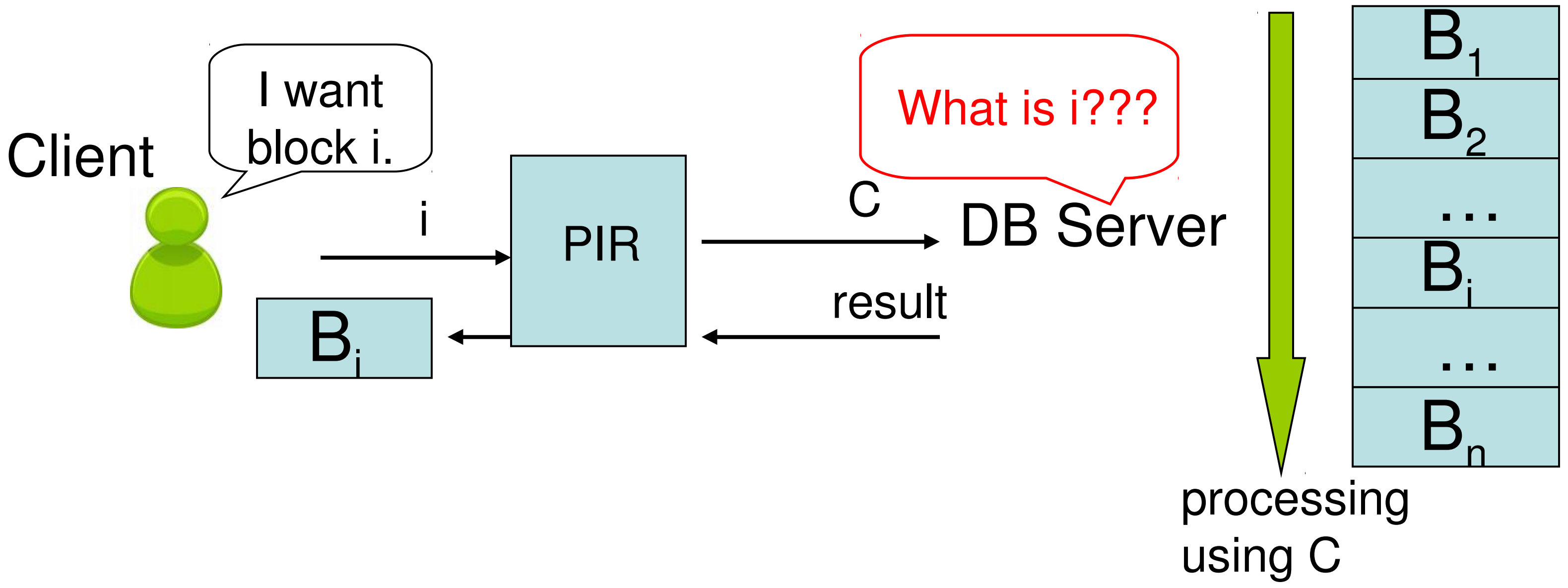
⇨ Access control

- File sharing in encrypted file systems
- Key distribution
- Encrypted mail to mailing lists
- Content protection (revoke compromised DVD players)

Private Information Retrieval (PIR)

PIR

- Retrieve item from a database without revealing to the database what item was retrieved



PIR (Cont'd)

- Naïve solution: send all database
  - O(n) bandwidth
- Current PIRs:
  - $(\log n)^2$ communication: *[Lipmaa, 2004], [Gentry and Ramzan, 2005]*
- Must touch all data blocks
- Implementation of best known PIR techniques:

  http://crypto.stanford.edu/pir-library/

Applications

- Privacy in databases: query unknown to the DB server
- Privacy in search

There are others..

- Blind signature schemes,
- Deniable encryption
- Proxy re-encryption
- Key rolling
- Ecash
- CS proofs
- Threshold decryption
- Secure-multi party computation