Disk Encryption

Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute, Kolkata
India
palash@isical.ac.in

Applied Statistics Unit
28th July, 2009



Computers





Fact. All computers store data on disks.

High Profile Users

Possessors of sensitive data.

- Heads of states.
- Important ministers.
- Top bureaucrats.
- Heads of armed forces.

Imagine the value of data on the disks of such personalities.



High Profile Users

Possessors of sensitive data.

- Heads of states.
- Important ministers.
- Top bureaucrats.
- Heads of armed forces.

Imagine the value of data on the disks of such personalities.



High Profile Users

Possessors of sensitive data.

- Heads of states.
- Important ministers.
- Top bureaucrats.
- Heads of armed forces.

Imagine the value of data on the disks of such personalities.



Threats

- What happens if a disk is lost?
- What happens if a disk is lost for some time?
  - The loss may be detected.
  - The loss may not be detected.
  - Should one still trust the data on the disk?
- The general question of whether to trust the data on the disk can be even more subtle.



Threats

- What happens if a disk is lost?
- What happens if a disk is lost for some time?
  - The loss may be detected.
  - The loss may not be detected.
  - Should one still trust the data on the disk?
- The general question of whether to trust the data on the disk can be even more subtle.



Threats

- What happens if a disk is lost?
- What happens if a disk is lost for some time?
  - The loss may be detected.
  - The loss may not be detected.
  - Should one still trust the data on the disk?
- The general question of whether to trust the data on the disk can be even more subtle.



Threats

- What happens if a disk is lost?
- What happens if a disk is lost for some time?
  - The loss may be detected.
  - The loss may not be detected.
  - Should one still trust the data on the disk?
- The general question of whether to trust the data on the disk can be even more subtle.



Threats

- What happens if a disk is lost?
- What happens if a disk is lost for some time?
  - The loss may be detected.
  - The loss may not be detected.
  - Should one still trust the data on the disk?
- The general question of whether to trust the data on the disk can be even more subtle.



Threats

- What happens if a disk is lost?
- What happens if a disk is lost for some time?
  - The loss may be detected.
  - The loss may not be detected.
  - Should one still trust the data on the disk?
- The general question of whether to trust the data on the disk can be even more subtle.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Modelling (Paranoid) Security

Progressively stronger security goals.

- The disk should not be accessible to an unauthorized user.
- It should not be possible to tamper the data.
- More formal goals.
  - The contents of the disk should appear “random” to an unauthorized user.
  - If an unauthorized user changes even a single bit, then after decryption, the entire disk sector should appear to be “random”.
- The above notions can be formalized in a precise manner.

Goal. Obtain methods which can be *proved* to be secure under a precise definition and under clearly stated (reasonable) assumptions.



Disk Encryption: Why and How?

- Any disk is physically divided into sectors.
- Each sector has a unique sector address.
- The disk encryption module is a piece of hardware that resides just above the disk controller.
- Disk encryption methods encrypt and decrypt on a sector-wise basis.
- Any data passing through and from the disk passes through the disk encryption module.
- A user has a secret key which is required for the disk encryption module to function.
- All data on the disk is stored in an encrypted form. There is no back-up of the clear data. So, if the key is lost, then the data becomes unintelligible.



Disk Encryption: Why and How?

- Any disk is physically divided into sectors.
- Each sector has a unique sector address.
- The disk encryption module is a piece of hardware that resides just above the disk controller.
- Disk encryption methods encrypt and decrypt on a sector-wise basis.
- Any data passing through and from the disk passes through the disk encryption module.
- A user has a secret key which is required for the disk encryption module to function.
- All data on the disk is stored in an encrypted form. There is no back-up of the clear data. So, if the key is lost, then the data becomes unintelligible.



Disk Encryption: Why and How?

- Any disk is physically divided into sectors.
- Each sector has a unique sector address.
- The disk encryption module is a piece of hardware that resides just above the disk controller.
- Disk encryption methods encrypt and decrypt on a sector-wise basis.
- Any data passing through and from the disk passes through the disk encryption module.
- A user has a secret key which is required for the disk encryption module to function.
- All data on the disk is stored in an encrypted form. There is no back-up of the clear data. So, if the key is lost, then the data becomes unintelligible.



Disk Encryption Module in Software

- Encryption and decryption done in main memory.
- Software implementations will typically be slower than hardware implementations.
- On the other hand, software based solutions are easier to implement.
- The software module can be placed at several places in the architecture.
  - At the device driver level.
  - At the file-system level.
- Implementations can be in high level language or in assembly.
  - Assembly implementation will be faster.



Disk Encryption Module in Software

- Encryption and decryption done in main memory.
- Software implementations will typically be slower than hardware implementations.
- On the other hand, software based solutions are easier to implement.
- The software module can be placed at several places in the architecture.
  - At the device driver level.
  - At the file-system level.
- Implementations can be in high level language or in assembly.
  - Assembly implementation will be faster.



Disk Encryption Module in Software

- Encryption and decryption done in main memory.
- Software implementations will typically be slower than hardware implementations.
- On the other hand, software based solutions are easier to implement.
- The software module can be placed at several places in the architecture.
  - At the device driver level.
  - At the file-system level.
- Implementations can be in high level language or in assembly.
  - Assembly implementation will be faster.



Disk Encryption Module in Software

- Encryption and decryption done in main memory.
- Software implementations will typically be slower than hardware implementations.
- On the other hand, software based solutions are easier to implement.
- The software module can be placed at several places in the architecture.
  - At the device driver level.
  - At the file-system level.
- Implementations can be in high level language or in assembly.
  - Assembly implementation will be faster.



Disk Encryption Module in Software

- Encryption and decryption done in main memory.
- Software implementations will typically be slower than hardware implementations.
- On the other hand, software based solutions are easier to implement.
- The software module can be placed at several places in the architecture.
  - At the device driver level.
  - At the file-system level.
- Implementations can be in high level language or in assembly.
  - Assembly implementation will be faster.



Disk Encryption Module in Software

- Encryption and decryption done in main memory.
- Software implementations will typically be slower than hardware implementations.
- On the other hand, software based solutions are easier to implement.
- The software module can be placed at several places in the architecture.
  - At the device driver level.
  - At the file-system level.
- Implementations can be in high level language or in assembly.
  - Assembly implementation will be faster.



The Cost of Security

- Encryption and decryption operations take time.
  - So, disk throughput degrades.
  - Goal: minimize this degradation.
- Disk encryption module is a piece of hardware.
  - Hardware requires space (and money).
  - For portable devices, space is at a premium.
  - Goal: minimize size of hardware.
- Cost of human resources.
  - Secure and efficient designs are hard to come by.
  - Implementations are not robust.
  - A small change can make a design insecure.



The Cost of Security

- Encryption and decryption operations take time.
  - So, disk throughput degrades.
  - Goal: minimize this degradation.
- Disk encryption module is a piece of hardware.
  - Hardware requires space (and money).
  - For portable devices, space is at a premium.
  - Goal: minimize size of hardware.
- Cost of human resources.
  - Secure and efficient designs are hard to come by.
  - Implementations are not robust.
  - A small change can make a design insecure.



The Cost of Security

- Encryption and decryption operations take time.
  - So, disk throughput degrades.
  - Goal: minimize this degradation.
- Disk encryption module is a piece of hardware.
  - Hardware requires space (and money).
  - For portable devices, space is at a premium.
  - Goal: minimize size of hardware.
- Cost of human resources.
  - Secure and efficient designs are hard to come by.
  - Implementations are not robust.
  - A small change can make a design insecure.



Disk Encryption: Solutions Without Guarantees

- A search on google will provide the names of several solutions.
- These are typically software based solutions.
  - Most of these are *adhoc* solutions.
  - Do not provide provable security guarantees.
  - Example: Bit locker cipher for Windows Vista (Ferguson 2006): superficial security analysis.



Disk Encryption: Formal Approach

- Encryption: $\mathbf{E}_K(T, M)$; Decryption: $\mathbf{D}_K(T, M)$.
- K: secret key;
  T: sector address (tweak);
  M: data to be stored in the sector.
- For every K and T, $D_K^T(E_K^T(M)) = M$.
- M is a fixed length bit string (usually 512 bytes).
- This is a special case of a more general definition of a length preserving permutation over variable length strings.



Disk Encryption: Formal Approach

- Encryption: $\mathbf{E}_K(T, M)$; Decryption: $\mathbf{D}_K(T, M)$.
- K: secret key;
  T: sector address (tweak);
  M: data to be stored in the sector.
- For every K and T, $D_K^T(E_K^T(M)) = M$.
- M is a fixed length bit string (usually 512 bytes).
- This is a special case of a more general definition of a length preserving permutation over variable length strings.



Disk Encryption: Formal Approach

- Encryption: $\mathbf{E}_K(T, M)$; Decryption: $\mathbf{D}_K(T, M)$.
- K: secret key;
  T: sector address (tweak);
  M: data to be stored in the sector.
- For every K and T, $D_K^T(E_K^T(M)) = M$.
- M is a fixed length bit string (usually 512 bytes).
- This is a special case of a more general definition of a length preserving permutation over variable length strings.



Disk Encryption: Security Model

- Choose a uniform random key K.
- Provide an adversary $\mathcal{A}$ with oracle (black-box) access to $\mathbf{E}_K(\cdot,\cdot)$ and $\mathbf{D}_K(\cdot,\cdot)$.
- At the end of the interaction, $\mathcal{A}$ produces a bit.
- Let $\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1$ be the event that $\mathcal{A}$ produces 1.
- Similarly, define $\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1$, where $\$(T, M)$ returns a uniform random string of length $|M|$.

$$\mathrm{Adv}^{\pm\mathrm{nrnd}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1].$$

$\mathrm{Adv}(\mathrm{ZZZ})$: maximum of $\mathrm{Adv}(\mathcal{A})$ over all adversaries $\mathcal{A}$ having resources ZZZ.



Disk Encryption: Security Model

- Choose a uniform random key K.
- Provide an adversary $\mathcal{A}$ with oracle (black-box) access to $\mathbf{E}_K(\cdot,\cdot)$ and $\mathbf{D}_K(\cdot,\cdot)$.
- At the end of the interaction, $\mathcal{A}$ produces a bit.
- Let $\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1$ be the event that $\mathcal{A}$ produces 1.
- Similarly, define $\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1$, where $\$(T, M)$ returns a uniform random string of length $|M|$.

$$\mathrm{Adv}^{\pm\mathrm{nrnd}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1].$$

$\mathrm{Adv}(\mathrm{ZZZ})$: maximum of $\mathrm{Adv}(\mathcal{A})$ over all adversaries $\mathcal{A}$ having resources ZZZ.



Disk Encryption: Security Model

- Choose a uniform random key K.
- Provide an adversary $\mathcal{A}$ with oracle (black-box) access to $\mathbf{E}_K(\cdot,\cdot)$ and $\mathbf{D}_K(\cdot,\cdot)$.
- At the end of the interaction, $\mathcal{A}$ produces a bit.
- Let $\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1$ be the event that $\mathcal{A}$ produces 1.
- Similarly, define $\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1$, where $\$(T, M)$ returns a uniform random string of length $|M|$.

$$\mathrm{Adv}^{\pm\mathrm{nrnd}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1].$$

$\mathrm{Adv}(\mathrm{ZZZ})$: maximum of $\mathrm{Adv}(\mathcal{A})$ over all adversaries $\mathcal{A}$ having resources ZZZ.



Disk Encryption: Security Model

- Choose a uniform random key K.
- Provide an adversary $\mathcal{A}$ with oracle (black-box) access to $\mathbf{E}_K(\cdot,\cdot)$ and $\mathbf{D}_K(\cdot,\cdot)$.
- At the end of the interaction, $\mathcal{A}$ produces a bit.
- Let $\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1$ be the event that $\mathcal{A}$ produces 1.
- Similarly, define $\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1$, where $\$(T, M)$ returns a uniform random string of length $|M|$.

$$\mathsf{Adv}^{\pm\mathsf{nrnd}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1].$$

$\mathsf{Adv}(\mathsf{ZZZ})$: maximum of $\mathsf{Adv}(\mathcal{A})$ over all adversaries $\mathcal{A}$ having resources ZZZ.



Disk Encryption: Security Model

- Choose a uniform random key K.
- Provide an adversary $\mathcal{A}$ with oracle (black-box) access to $\mathbf{E}_K(\cdot,\cdot)$ and $\mathbf{D}_K(\cdot,\cdot)$.
- At the end of the interaction, $\mathcal{A}$ produces a bit.
- Let $\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1$ be the event that $\mathcal{A}$ produces 1.
- Similarly, define $\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1$, where $\$(T, M)$ returns a uniform random string of length $|M|$.

$$\mathrm{Adv}^{\pm\mathrm{nrnd}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot),\mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot),\$(\cdot,\cdot)} \Rightarrow 1].$$

$\mathrm{Adv}(\mathrm{ZZZ})$: maximum of $\mathrm{Adv}(\mathcal{A})$ over all adversaries $\mathcal{A}$ having resources ZZZ.



Patented and Standardized Constructions

Patents.

- Cisco Systems. Extended Codebook Mode (XCB).
  US Patent 7418100, issued on August 26, 2008.
- University of California. Encrypt-Mix-Encrypt (EME).
  US Patent Application 20040131182A1 from the University of California.

Standards.

- IEEE Security in Storage Working Group:
  P1619.2 (Wide-Block Encryption).
  As of November 2008: decision to standardize only XCB and EME2.



Patented and Standardized Constructions

Patents.

- Cisco Systems. Extended Codebook Mode (XCB).
  US Patent 7418100, issued on August 26, 2008.
- University of California. Encrypt-Mix-Encrypt (EME).
  US Patent Application 20040131182A1 from the University of California.

Standards.

- IEEE Security in Storage Working Group:
  P1619.2 (Wide-Block Encryption).
  As of November 2008: decision to standardize only XCB and EME2.



New Constructions

Sarkar 2008-09.

Features.

- Faster than all previously known schemes.
  - Efficiency improvements in both software and hardware.
- Can be implemented in significantly less hardware.
- More efficient key agility.
- No IP claims.



Basic Components

- Let π be a uniform random permutation of $\{0,1\}^n$.
  - In practice π will be instantiated by a block cipher.

$$E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$$

  - **key space**: $\mathcal{K} = \{0,1\}^k$.
  - **message and cipher spaces**: $\mathcal{M} = \{0,1\}^n, \mathcal{C} = \{0,1\}^n$.
  - $E_K(M) = E(K, M)$ is a permutation for every fixed K.
  - Typical values: $n, k = 128, 192, 256$, with $k \geq n$.
- Universal hash function: for $x \neq x'$ and uniform random τ,

$$\Pr_\tau[\mathcal{F}_\tau(x) = \mathcal{F}_\tau(x')] \leq \varepsilon.$$

Several variants are actually used.



Basic Components

- Let π be a uniform random permutation of $\{0,1\}^n$.
  - In practice π will be instantiated by a block cipher.

$$E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$$

  - **key space**: $\mathcal{K} = \{0,1\}^k$.
  - **message and cipher spaces**: $\mathcal{M} = \{0,1\}^n, \mathcal{C} = \{0,1\}^n$.
  - $E_K(M) = E(K, M)$ is a permutation for every fixed K.
  - Typical values: $n, k = 128, 192, 256$, with $k \geq n$.
- Universal hash function: for $x \neq x'$ and uniform random τ,

$$\Pr_\tau[\mathcal{F}_\tau(x) = \mathcal{F}_\tau(x')] \leq \varepsilon.$$

Several variants are actually used.



Basic Components

- Let π be a uniform random permutation of $\{0,1\}^n$.
  - In practice π will be instantiated by a block cipher.

$$E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$$

  - **key space**: $\mathcal{K} = \{0,1\}^k$.
  - **message and cipher spaces**: $\mathcal{M} = \{0,1\}^n$, $\mathcal{C} = \{0,1\}^n$.
  - $E_K(M) = E(K, M)$ is a permutation for every fixed K.
  - Typical values: $n, k = 128, 192, 256$, with $k \geq n$.
- Universal hash function: for $x \neq x'$ and uniform random τ,

$$\Pr_\tau[\mathcal{F}_\tau(x) = \mathcal{F}_\tau(x')] \leq \varepsilon.$$

Several variants are actually used.



Universal Hashing

- $$\mathcal{F}_\tau(P_1, \ldots, P_m) = P_m + \tau P_{m-1} + \cdots + \tau^{m-1} P_1.$$
- Requires $(m-1)$ multplications; $m/|\mathbb{F}|$-AU; $\tau\mathcal{F}_\tau$ satisfies XOR universality.
- Bernstein (2007): introduces a class of polynomials which builds upon earlier work by Rabin-Winograd;
- BRW polynomials can be computed using $m/2$ multiplications; $2m/|\mathbb{F}|$-AU;
- Sarkar (2008): universal hashing from word oriented LFSRs;
- no multiplications; bitwise processing – slow in software; good for resource constrained devices.



Universal Hashing

- $$\mathcal{F}_\tau(P_1, \ldots, P_m) = P_m + \tau P_{m-1} + \cdots + \tau^{m-1} P_1.$$
- Requires $(m-1)$ multplications; $m/|\mathbb{F}|$-AU; $\tau\mathcal{F}_\tau$ satisfies XOR universality.
- Bernstein (2007): introduces a class of polynomials which builds upon earlier work by Rabin-Winograd;
- BRW polynomials can be computed using $m/2$ multiplications; $2m/|\mathbb{F}|$-AU;
- Sarkar (2008): universal hashing from word oriented LFSRs;
- no multiplications; bitwise processing – slow in software; good for resource constrained devices.



Universal Hashing

- $$\mathcal{F}_\tau(P_1, \ldots, P_m) = P_m + \tau P_{m-1} + \cdots + \tau^{m-1} P_1.$$
- Requires $(m-1)$ multplications; $m/|\mathbb{F}|$-AU; $\tau\mathcal{F}_\tau$ satisfies XOR universality.
- Bernstein (2007): introduces a class of polynomials which builds upon earlier work by Rabin-Winograd;
- BRW polynomials can be computed using $m/2$ multiplications; $2m/|\mathbb{F}|$-AU;
- Sarkar (2008): universal hashing from word oriented LFSRs;
- no multiplications; bitwise processing – slow in software; good for resource constrained devices.



Invertible Blockwise Universal

- Fix integer $m \geq 1$ and let $\mathbb{F}$ be a finite field.

$$\mathcal{F} : \mathcal{K} \times \mathbb{F}^m \rightarrow \mathbb{F}^m$$

  - For each $K \in \mathcal{K}$, $\mathcal{F}_K$ is invertible.
- ϵ-BAU: for $(\mathbf{x}, i) \neq (\mathbf{x}', i')$, $1 \leq i, i' \leq m$.

$$\begin{aligned} \mathcal{F}_K : & \quad (X_1, \ldots, X_m) \mapsto (Y_1, \ldots, Y_m) \\ \mathcal{F}_K : & \quad (X'_1, \ldots, X'_m) \mapsto (Y'_1, \ldots, Y'_m) \end{aligned}$$

$$\Pr_K[Y_i = Y'_{i'}] \leq \epsilon.$$



Invertible Blockwise Universal

- Fix integer $m \geq 1$ and let $\mathbb{F}$ be a finite field.

$$\mathcal{F} : \mathcal{K} \times \mathbb{F}^m \rightarrow \mathbb{F}^m$$

  - For each $K \in \mathcal{K}$, $\mathcal{F}_K$ is invertible.
- ϵ-BAU: for $(\mathbf{x}, i) \neq (\mathbf{x}', i')$, $1 \leq i, i' \leq m$.

$$\begin{array}{rccc} \mathcal{F}_K : & (X_1, \ldots, X_m) & \mapsto & (Y_1, \ldots, Y_m) \\ \mathcal{F}_K : & (X'_1, \ldots, X'_m) & \mapsto & (Y'_1, \ldots, Y'_m) \end{array}$$

$$\Pr_K[Y_i = Y'_{i'}] \leq \epsilon.$$



Invertible Blockwise Universal Hash Constructions

- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus u(Y), \ldots, X_{m-1} \oplus u(Y), Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (NR 99)
  - $(X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (Sarkar 08)

  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$; ϕ, ψ and u are AXU;
- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus Z, \ldots, X_m \oplus Z) \oplus (\beta, \alpha\beta, \ldots, \alpha^{m-1}\beta)$ (Halevi 07)

  $Z = \sigma^{-1}(X_1\tau^m \oplus \cdots \oplus X_{m-1}\tau^2 \oplus X_m\tau)$; $\sigma = 1 \oplus \tau \oplus \cdots \oplus \tau^m$.



Invertible Blockwise Universal Hash Constructions

- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus u(Y), \ldots, X_{m-1} \oplus u(Y), Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (NR 99)
  - $(X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (Sarkar 08)

  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$; ϕ, ψ and u are AXU;
- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus Z, \ldots, X_m \oplus Z) \oplus (\beta, \alpha\beta, \ldots, \alpha^{m-1}\beta)$ (Halevi 07)

  $Z = \sigma^{-1}(X_1\tau^m \oplus \cdots \oplus X_{m-1}\tau^2 \oplus X_m\tau)$; $\sigma = 1 \oplus \tau \oplus \cdots \oplus \tau^m$.



Invertible Blockwise Universal Hash Constructions

- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus u(Y), \ldots, X_{m-1} \oplus u(Y), Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (NR 99)
  - $(X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (Sarkar 08)

  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$; ϕ, ψ and u are AXU;
- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus Z, \ldots, X_m \oplus Z) \oplus (\beta, \alpha\beta, \ldots, \alpha^{m-1}\beta)$ (Halevi 07)

  $Z = \sigma^{-1}(X_1\tau^m \oplus \cdots \oplus X_{m-1}\tau^2 \oplus X_m\tau)$; $\sigma = 1 \oplus \tau \oplus \cdots \oplus \tau^m$.



Invertible Blockwise Universal Hash Constructions

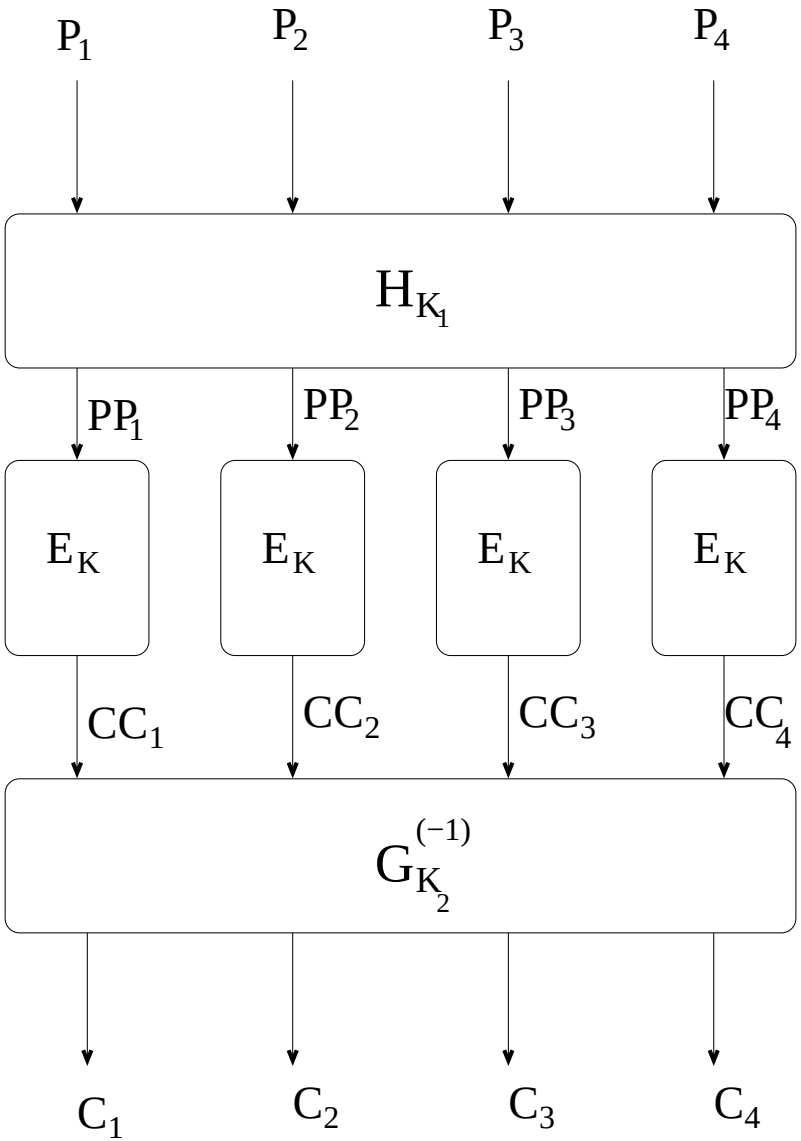
- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus u(Y), \ldots, X_{m-1} \oplus u(Y), Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (NR 99)
  - $(X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus (\phi_\beta(0), \ldots, \phi_\beta(m-1))$ (Sarkar 08)

  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$; ϕ, ψ and u are AXU;
- $(X_1, \ldots, X_m) \mapsto$
  - $(X_1 \oplus Z, \ldots, X_m \oplus Z) \oplus (\beta, \alpha\beta, \ldots, \alpha^{m-1}\beta)$ (Halevi 07)

  $Z = \sigma^{-1}(X_1\tau^m \oplus \cdots \oplus X_{m-1}\tau^2 \oplus X_m\tau)$; $\sigma = 1 \oplus \tau \oplus \cdots \oplus \tau^m$.



Hash-ECB-Hash



H and *G* are invertible blockwise universal hash functions.

HEH (Full Blocks)

- $$H_{K_1} = \Psi_{\tau,\beta_1}, \quad G_{K_2} = \Psi_{\tau,\beta_2}.$$

- $\Psi_{\tau,\beta}(X_1, \ldots, X_m)$
  $= (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus$
  $(\phi_\beta(1), \ldots, \phi_\beta(m-1), \phi_\beta(0))$
  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$.
- ϕ_β:
  - $i \mapsto \alpha^i \beta$; α is a primitive element of $GF(2^n)$ ("powering map").
  - Using a tower field representation of $GF(2^n)$, it is possible to obtain a faster implementation of ϕ (Sarkar 2008).
- ψ_τ can be instantiated using
  - usual polynomial hashing;
  - hashing using BRW polynomials;
  - hashing using word oriented LFSRs.



HEH (Full Blocks)

- $$H_{K_1} = \Psi_{\tau,\beta_1}, \quad G_{K_2} = \Psi_{\tau,\beta_2}.$$

- $\Psi_{\tau,\beta}(X_1, \ldots, X_m)$
  $= (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus$
  $(\phi_\beta(1), \ldots, \phi_\beta(m-1), \phi_\beta(0))$
  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$.
- ϕ_β:
  - $i \mapsto \alpha^i \beta$; α is a primitive element of $GF(2^n)$ ("powering map").
  - Using a tower field representation of $GF(2^n)$, it is possible to obtain a faster implementation of ϕ (Sarkar 2008).
- ψ_τ can be instantiated using
  - usual polynomial hashing;
  - hashing using BRW polynomials;
  - hashing using word oriented LFSRs.



HEH (Full Blocks)

- $$H_{K_1} = \Psi_{\tau,\beta_1}, \quad G_{K_2} = \Psi_{\tau,\beta_2}.$$

- $\Psi_{\tau,\beta}(X_1, \ldots, X_m)$
  $= (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus$
  $(\phi_\beta(1), \ldots, \phi_\beta(m-1), \phi_\beta(0))$
  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$.
- ϕ_β:
  - $i \mapsto \alpha^i \beta$; α is a primitive element of $GF(2^n)$ ("powering map").
  - Using a tower field representation of $GF(2^n)$, it is possible to obtain a faster implementation of ϕ (Sarkar 2008).
- ψ_τ can be instantiated using
  - usual polynomial hashing;
  - hashing using BRW polynomials;
  - hashing using word oriented LFSRs.



HEH (Full Blocks)

- $$H_{K_1} = \Psi_{\tau,\beta_1}, \quad G_{K_2} = \Psi_{\tau,\beta_2}.$$

- $\Psi_{\tau,\beta}(X_1, \ldots, X_m)$
  $= (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus$
  $(\phi_\beta(1), \ldots, \phi_\beta(m-1), \phi_\beta(0))$
  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$.
- ϕ_β:
  - $i \mapsto \alpha^i \beta$; α is a primitive element of $GF(2^n)$ ("powering map").
  - Using a tower field representation of $GF(2^n)$, it is possible to obtain a faster implementation of ϕ (Sarkar 2008).
- ψ_τ can be instantiated using
  - usual polynomial hashing;
  - hashing using BRW polynomials;
  - hashing using word oriented LFSRs.



HEH (Full Blocks)

- $$H_{K_1} = \Psi_{\tau,\beta_1}, \quad G_{K_2} = \Psi_{\tau,\beta_2}.$$

- $\Psi_{\tau,\beta}(X_1, \ldots, X_m)$
  $= (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus$
  $(\phi_\beta(1), \ldots, \phi_\beta(m-1), \phi_\beta(0))$
  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$.
- ϕ_β:
  - $i \mapsto \alpha^i \beta$; α is a primitive element of $GF(2^n)$ ("powering map").
  - Using a tower field representation of $GF(2^n)$, it is possible to obtain a faster implementation of ϕ (Sarkar 2008).
- ψ_τ can be instantiated using
  - usual polynomial hashing;
  - hashing using BRW polynomials;
  - hashing using word oriented LFSRs.



HEH (Full Blocks)

- $$H_{K_1} = \Psi_{\tau,\beta_1}, \quad G_{K_2} = \Psi_{\tau,\beta_2}.$$

- $\Psi_{\tau,\beta}(X_1, \ldots, X_m)$
  $= (X_1 \oplus Y, \ldots, X_{m-1} \oplus Y, Y) \oplus$
  $(\phi_\beta(1), \ldots, \phi_\beta(m-1), \phi_\beta(0))$
  $Y = X_m \oplus \psi_\tau(X_1, \ldots, X_{m-1})$.
- ϕ_β:
  - $i \mapsto \alpha^i \beta$; α is a primitive element of $GF(2^n)$ ("powering map").
  - Using a tower field representation of $GF(2^n)$, it is possible to obtain a faster implementation of ϕ (Sarkar 2008).
- ψ_τ can be instantiated using
  - usual polynomial hashing;
  - hashing using BRW polynomials;
  - hashing using word oriented LFSRs.



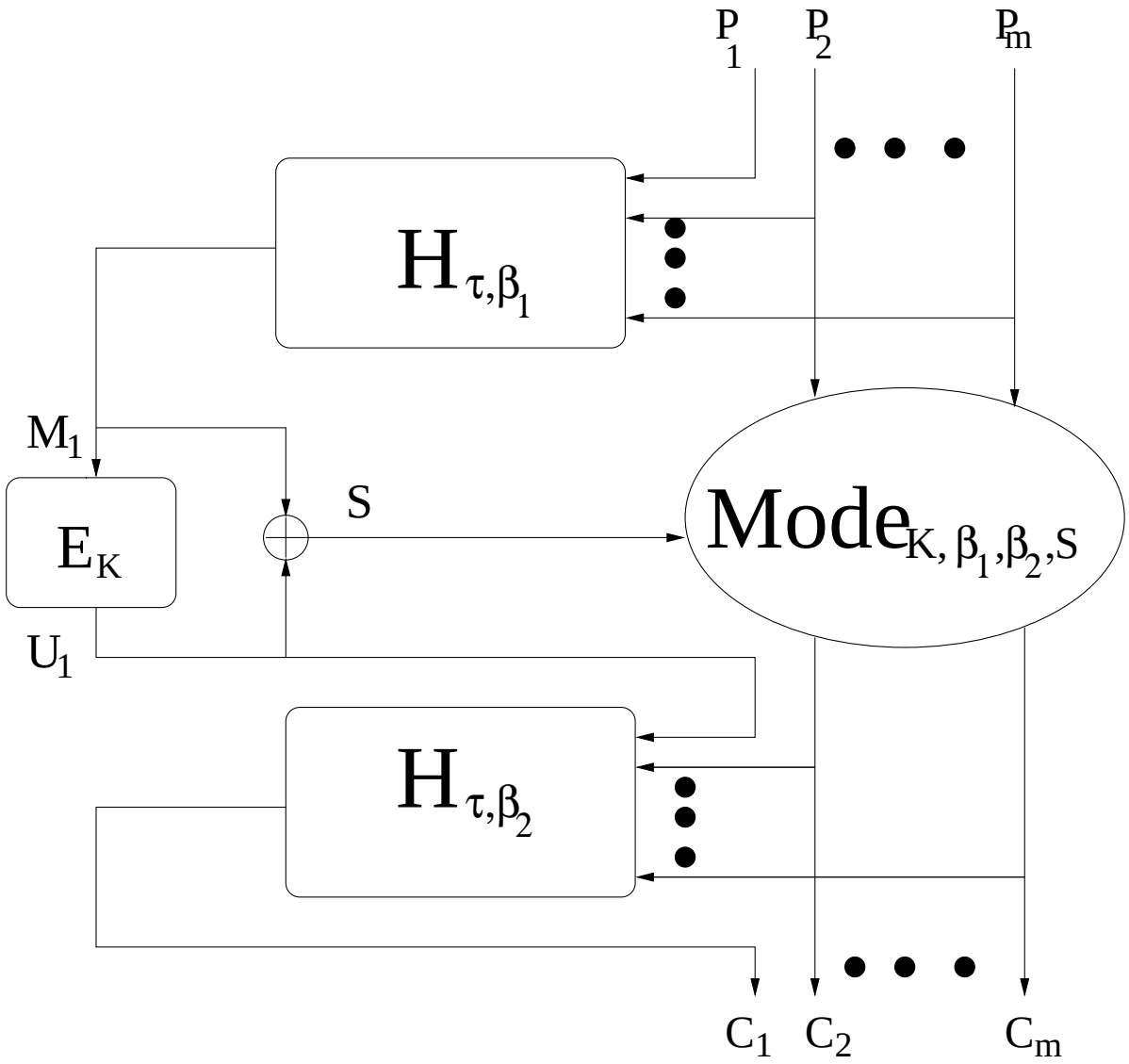
Definition of τ, β_1, β_2

KeyDef3: for fixed length messages (disk encryption).

KeyDef3
$\beta_1 = E_K(T)$; $\beta_2 = \alpha\beta_1$;   choose τ randomly   from $GF(2^n)$.



HMCH/HOH Modes



HMCH/HOH Modes

$$H_{\tau,\beta}(X_1, \ldots, X_m) = \beta \oplus X_1 \oplus \psi_\tau(X_2, \ldots, X_m).$$

- $\text{Mode}_{K,\beta_1,\beta_2,S}$ as Counter:

$$\begin{aligned} \text{Ctr}_{K,\beta_1,R}(X_1, \ldots, X_m) &= \text{ECB}_K(R_1, \ldots, R_m) \oplus (X_1, \ldots, X_m) \\ &= (E_K(R_1), \ldots, E_K(R_m)) \oplus (X_1, \ldots, X_m) \end{aligned}$$

  where $R = S \oplus \beta_1 \oplus \beta_2$, $R_i = \phi_\beta(i-1) \oplus R$.

- $\text{Mode}_{K,\beta_1,\beta_2,S}$ as OFB:

$$\begin{aligned} &\text{OFB}_{K,S}(X_1, \ldots, X_m) \\ &\quad = (X_1, \ldots, X_m) \oplus (S_1, \ldots, S_m) \end{aligned}$$

  where $S_i = E_K^i(S)$; β_1 and β_2 are not used.



HMCH/HOH Modes

$$H_{\tau,\beta}(X_1, \ldots, X_m) = \beta \oplus X_1 \oplus \psi_\tau(X_2, \ldots, X_m).$$

- $\text{Mode}_{K,\beta_1,\beta_2,S}$ as Counter:

$$\begin{aligned} \text{Ctr}_{K,\beta_1,R}(X_1, \ldots, X_m) &= \text{ECB}_K(R_1, \ldots, R_m) \oplus (X_1, \ldots, X_m) \\ &= (E_K(R_1), \ldots, E_K(R_m)) \oplus (X_1, \ldots, X_m) \end{aligned}$$

  where $R = S \oplus \beta_1 \oplus \beta_2$, $R_i = \phi_\beta(i-1) \oplus R$.

- $\text{Mode}_{K,\beta_1,\beta_2,S}$ as OFB:

$$\begin{aligned} &\text{OFB}_{K,S}(X_1, \ldots, X_m) \\ &\quad = (X_1, \ldots, X_m) \oplus (S_1, \ldots, S_m) \end{aligned}$$

  where $S_i = E_K^i(S)$; β_1 and β_2 are not used.



Security

Theorem. Fix n and σ_n to be positive integers. Suppose that an adversary uses a total of σ_n blocks in all its queries, where each block is an n-bit string. Then

$$\mathbf{Adv}^{\pm\text{nrnd}}_{\text{XXX}[\text{Perm}(n)]}(\sigma_n) \leq \frac{20\sigma_n^2}{2^n}$$

Here XXX is one of the tweakable enciphering schemes HEH, HMCH or HOH.

- Security analysis is information theoretic, i.e., done using π rather than E_K.
- Computational security follows in a standard manner:
  - an additive degradation by the advantage of an adversary in distinguishing (π, π^{-1}) from (E_K, E_K^{-1}).



Security

Theorem. Fix n and σ_n to be positive integers. Suppose that an adversary uses a total of σ_n blocks in all its queries, where each block is an n-bit string. Then

$$\mathbf{Adv}^{\pm\text{nrnd}}_{\text{XXX}[\text{Perm}(n)]}(\sigma_n) \leq \frac{20\sigma_n^2}{2^n}$$

Here XXX is one of the tweakable enciphering schemes HEH, HMCH or HOH.

- Security analysis is information theoretic, i.e., done using π rather than E_K.
- Computational security follows in a standard manner:
  - an additive degradation by the advantage of an adversary in distinguishing (π, π^{-1}) from (E_K, E_K^{-1}).



Security

Theorem. Fix n and σ_n to be positive integers. Suppose that an adversary uses a total of σ_n blocks in all its queries, where each block is an n-bit string. Then

$$\mathbf{Adv}^{\pm\text{nrnd}}_{\text{XXX}[\text{Perm}(n)]}(\sigma_n) \leq \frac{20\sigma_n^2}{2^n}$$

Here XXX is one of the tweakable enciphering schemes HEH, HMCH or HOH.

- Security analysis is information theoretic, i.e., done using π rather than E_K.
- Computational security follows in a standard manner:
  - an additive degradation by the advantage of an adversary in distinguishing (π, π^{-1}) from (E_K, E_K^{-1}).



Comparison Issues

Security: comparison meaningful between modes providing the same security level.

Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Comparison Issues

Security: comparison meaningful between modes providing the same security level.
Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Comparison Issues

Security: comparison meaningful between modes providing the same security level.
Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Comparison Issues

Security: comparison meaningful between modes providing the same security level.
Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Comparison Issues

Security: comparison meaningful between modes providing the same security level.
Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Comparison Issues

Security: comparison meaningful between modes providing the same security level.
Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Comparison Issues

Security: comparison meaningful between modes providing the same security level.
Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Comparison Issues

Security: comparison meaningful between modes providing the same security level.
Efficiency:

- Number of block cipher calls.
- Number of finite field multiplications (if any).
- Number of block cipher keys.
- Amount of other key material (if any).
- Use of pre-computed multiplication tables (if relevant).
- Parallelism, simplicity, ...



Fixed Length Messages: Disk Encryption

Mode	[BC]	[M]	[BCK]	[AK]
EME	$2m+1$	–	1	2
XCB	$m+1$	$2(m+3)$	3	2
TET	$m+1$	$2m$	2	3
XXX	$m+1$	m	1	1

XXX: HEH, HMCH, HOH with BRW polynomials; for EME $m < n$.

Typical values: $n = 128$, $m = 32$ (corresponding to a 512-byte sector).



Key Agility

Mode	comp. cost	key sch.
EME	–	1
XCB	5[BC]	3
TET	$\imath((m-1)$[M]+2[BC])+ 1[BC]+1[I]	2
XXX	–	1

XXX: HEH, HMCH, HOH; $\imath \geq 1$.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Comparison Summary

- Hardware implementation:
  - HEH, HMCH, HOH are clearly better than EME, XCB and TET.
  - FPGA implementations by Mexico-based researchers confirm the above.
  - Issues: chip size, memory, ...
  - Using multiplication-less universal hash function (Sarkar 2008) should reduce hardware size. Not yet implemented.
- Software implementation:
  - HOH will be the fastest: there are no masking operations; HEH, HMCH are also good choices;
  - XCB has less efficient key agility;

    TET: really bad key agility.
  - EME is OK; EMME (Sarkar 2008) is slightly better.



Subsequent Work

- All previously known disk encryption schemes require both $E_K()$ and $E_K^{-1}()$.
- In a recent work, we have described a scheme which requires only $E_K()$.
  - This results in significant savings in hardware implementation.
- We have used similar ideas to develop schemes which use stream ciphers with IV instead of block ciphers.
  - Can be used with hardware oriented stream ciphers.
  - Suitable for very small devices (thumb disk).



Subsequent Work

- All previously known disk encryption schemes require both $E_K()$ and $E_K^{-1}()$.
- In a recent work, we have described a scheme which requires only $E_K()$.
  - This results in significant savings in hardware implementation.
- We have used similar ideas to develop schemes which use stream ciphers with IV instead of block ciphers.
  - Can be used with hardware oriented stream ciphers.
  - Suitable for very small devices (thumb disk).



Subsequent Work

- All previously known disk encryption schemes require both $E_K()$ and $E_K^{-1}()$.
- In a recent work, we have described a scheme which requires only $E_K()$.
  - This results in significant savings in hardware implementation.
- We have used similar ideas to develop schemes which use stream ciphers with IV instead of block ciphers.
  - Can be used with hardware oriented stream ciphers.
  - Suitable for very small devices (thumb disk).



Subsequent Work

- All previously known disk encryption schemes require both $E_K()$ and $E_K^{-1}()$.
- In a recent work, we have described a scheme which requires only $E_K()$.
  - This results in significant savings in hardware implementation.
- We have used similar ideas to develop schemes which use stream ciphers with IV instead of block ciphers.
  - Can be used with hardware oriented stream ciphers.
  - Suitable for very small devices (thumb disk).



Thank you for your attention!

