iDecrypt

(EyeDecrypt)

Andrea Forte (AT&T Security Research Center)
Juan Garay (AT&T Labs — Research)
Yevgeniy Vahlis (AT&T Security Research Center)
“Shoulder Surfing”
(Traditional) Content Protection

- Encryption, authentication, selective decryption
- Receiver/display devices
This Work

- Content protection in new setting: public-view rendering device (next slide)
- Content can be stored/offline or dynamically captured (streaming)
- Two main components:
  - *Visualizable* encryption scheme
  - New visual encoding technique

“For your eyes only!”
Model

Content repository/ “capturing” device

\[ c = E_K(m) \]

\[ c' = E_K(m') \]

Non-malleability

(R(m,m'))

(Public) Rendering device
iDecrypt in a Nutshell


Visualizable Encryption

Visual Encoding

Plaintext

Ciphertext

Visual Encoding
Rest of the Talk

- Introduction/Motivation
- Visualizable Encryption
- Visual Encoding
- “Demo”
Rest of the Talk

- Introduction/Motivation
- Visualizable Encryption
- Visual Encoding
- “Demo”
**Visualizable Encryption**

- Plaintext space is a matrix of text (for now)

- Encryption is performed per block:
  \[
  \text{Enc}_{K_{doc}}(M, i, j) = (\text{PRF}_{K_{doc}}^{\text{enc}}(i, j) \oplus M, \tau, i, j)
  \]
  where \( \tau = \text{MAC}_{K_{doc}}^{\text{mac}}(\text{PRF}_{K_{doc}}^{\text{enc}}(i, j) \oplus M) \)

- Decryption is straightforward:
  \[
  \text{Dec}_{K_{doc}}(C_{i, j}, \tau, i, j) = C_{i, j} \oplus \text{PRF}_{K_{doc}}^{\text{enc}}(i, j)
  \]
  if \( \tau \) is correct and \( \perp \) otherwise
Visualizable Encryption (cont’d)

- “Frames” — allow dynamically changing content (e.g., video, screen encryption)
- Encryption is performed per block:
  $$\text{Enc}_{K_{doc}}(M, i, j) = (\text{PRF}_{K_{doc}^{enc}}(i, j, f) \oplus M, \tau, i, j, f)$$
  where $$\tau = \text{MAC}_{K_{doc}^{mac}}(\text{PRF}_{K_{doc}^{enc}}(i, j, f) \oplus M)$$

- Decryption is straightforward:
  $$\text{Dec}_{K_{doc}}(C_{i,j}, \tau, i, j, f) = C_{i,j} \oplus \text{PRF}_{K_{doc}^{enc}}(i, j, f)$$
  if $$\tau$$ is correct and $$\perp$$ otherwise
Visualizable Encryption: Security Definition

Vis-IND-CCA security: A visualizable encryption scheme is secure if no efficient Adv can win the following game:

1. **Setup:** The challenger chooses a random document key $K_{doc}$
2. **Query Phase 1:** Adv submits encryption and decryption queries $Enc(m,i,j)$ and $Dec(c,i,j)$
   - We restrict Adv from submitting $Enc(m,i,j)$ and $Enc(m',i,j)$ for $m \neq m'$
3. **Challenge:** The adversary submits $(m_0,m_1,i^*,j^*)$ that were not previously queried, and the challenger returns $c^* = Enc(m_b,i^*,j^*)$ for a random $b$
4. **Query Phase 2:** Same as phase 1, except Adv cannot query $Dec(c^*,i^*,j^*)$
5. **Guess:** Adv outputs a guess $b'$ for the value of $b$
Visualizable Encryption: Security Def. (cont’d)

**Vis-IND-CCA security:** A visualizable encryption scheme is secure if no efficient Adv can win the following game:

1. **Setup:** The challenger chooses a random document key $K_{\text{doc}}$
2. **Query Phase 1:** Adv submits encryption and decryption queries $\text{Enc}(m, i, j, f)$ and $\text{Dec}(c, i, j, f)$
   - We restrict Adv from submitting $\text{Enc}(m, i, j, f)$ and $\text{Enc}(m', i, j, f)$ for $m \neq m'$
3. **Challenge:** The adversary submits $(m_0, m_1, i^*, j^*, f^*)$ that were not previously queried, and the challenger returns $c^* = \text{Enc}(m_b, i^*, j^*, f^*)$ for a random $b$
4. **Query Phase 2:** Same as phase 1, except Adv cannot query $\text{Dec}(c^*, i^*, j^*, f^*)$
5. **Guess:** Adv outputs a guess $b'$ for the value of $b$
Visualizable Encryption: Proof Sketch

- Replacing PRF with random function assigns a random pad to each block coordinates
- Each block can be queried at most once to the encryption oracle → encrypted content is indistinguishable from random
- Non-malleability follows from MAC
Rest of the Talk

- Introduction/Motivation
- Visualizable Encryption
- Visual Encoding
- “Demo”
Visual Encoding

- Many existing visual encoding solutions: QR, data matrix, Dataglyphs, HCCB
- Most require capturing the entire encoding
- We require:
  - *Locality* – cropped encoding decodes to sub-matrix of input
  - *Relative positioning* – adjacent input sub-matrixes are adjacent in encoded image
Visual Encoding Scheme

- New encoding designed to meet our needs
- Bits are encoded as different-size blocks
- Distinguishing is easy regardless of distance/scale
- To encode a matrix with n-bit cells, compute an $\sqrt{n} \times \sqrt{n}$ bit matrix for each cell in input matrix
- Two colors are used to distinguish rows
- Bits can be read regardless of orientation or skew — “orientation robustness”
Visual Encoding Scheme (cont’d)

d₁, d₂ – dimensions of input matrix
t₁, t₂ – dimensions of output image
P – pixel space (e.g., RGB triples)

Definition of visual encoding:

\[
Enc: \{0, 1\}^{n \times d_1 \times d_2} \rightarrow P^{t_1 \times t_2}
\]

\[
Dec \overset{\text{def}}{=} Enc^{-1}
\]

Relative positioning:

\[
[r_1 \ldots r_2, c_1 \ldots c_2] \rightarrow [r_1 \cdot t_1/d_1 \ldots r_2 \cdot t_1/d_1, c_1 \cdot t_2/d_2 \ldots c_2 \cdot t_2/d_2]
\]
Visual Encoding Scheme (cont’d)

Decoding leverages alternating row colors and different block sizes

- First, “clean up” the image
- While reading a row, look for the closest unvisited block of the same color (e.g., using BFS)
- No need to untilt/unskew image (“orientation robustness”)
- To distinguish 0/1 compare relative areas of blocks
Rest of the Talk

- Introduction/Motivation
- Visualizable Encryption
- Visual Encoding
- “Demo”
“Demo”

1. Get Image from camera
2. Convert pixels values to (r,g,b)
3. Use BFS to find first / next block in a line
4. Use Flood Fill to compute blocks area
5. Display plaintext overlayd on camera view
6. Decrypt ciphertext
7. Build binary matrix (ciphertext)

Small block: 1
Large block: 0
“Demo”
“Demo”
“Demo”
“Demo”
Performance and Challenges

- **Capacity**
  - Plaintext: 5184 bits
  - Ciphertext: 7200 bits (overhead due to coordinates and padding)
  - Visual encoding: 1 bit / 0.25 cm²

- **Desired properties**
  - High(er) capacity visual encoding
  - Decoding possible for different resolutions (distance, camera)
    - Can include encryption of low resolution version of neighboring blocks

- **Performance**
  - Used phone (Samsung Galaxy S2) ’s GPU for faster rendering
  - Various algorithmic optimizations

- **Security definition(s)**
Thanks!