# Testable Commitments

Philippe Golle
Palo Alto Research Center
pgolle@parc.com

Richard Chow
Palo Alto Research Center
rchow@parc.com

Jessica Staddon
Palo Alto Research Center
staddon@parc.com

## ABSTRACT

A key challenge in litigation is verifying that all relevant case content has been produced. Adding to the challenge is the fact that litigating parties are both bound to produce relevant documents and bound to protect private information (e.g. medical information). This leaves open the possibility of withholding content inappropriately, and verifying that this has not occurred is a time-consuming process involving the presiding judge. We introduce testable commitments: a cryptographic technique for verifying that only the right information has been withheld with only minimal involvement from a trusted third party. We present a construction of testable commitments and discuss its implementation.

## Categories and Subject Descriptors

K.6.5 [**Computing Milieux**]: Management of Computing and Information Systems—*Security and Protection*

## General Terms

Security

## Keywords

Commitment scheme, privacy, discovery, responsive content, litigation.

## 1. INTRODUCTION

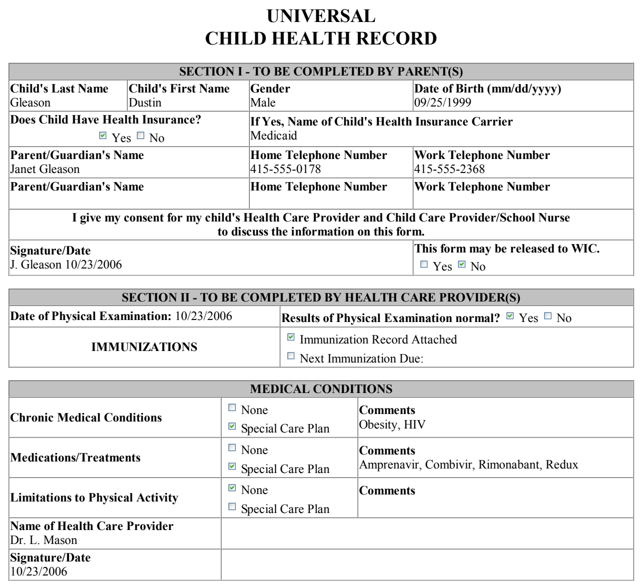
Discovery is the process by which opposing parties in litigation agree on which documents are relevant, or "responsive", to their case. This process is time-consuming, as it typically requires the review of very large collections of documents. The process is also often contentious, as parties may disagree over what documents to disclose and may fight to protect the privacy of their own documents, while trying to obtain as much information as possible from the opposing party. It is vitally important that the parties establish a credible process for producing responsive content, and withholding non-responsive content, as flaws in this process are grounds for appeal (see, for example, [10]).

The discovery process is overseen by a judge, who is trusted by both parties. The existing process is burdensome for the judge. Whenever a dispute arises over the responsiveness of a document, or portion of a document, the judge must review it to rule on its responsiveness. These disputes may arise over portions of documents as short as a paragraph, a sentence or even a single word that may have been redacted (i.e. blacked out) prior to production of the document. Redaction is common: its uses range from legislative compliance to privacy protection or the protection of information subject to attorney-client privilege. To rule on the appropriateness of redaction, the judge may have to resort to line-by-line comparisons between the original document and the redacted one. This manual review process is tedious, expensive and error-prone.

We introduce a new method of resolving disputes over the responsiveness of documents, which we call *testable commitments.* Testable commitment schemes extend traditional cryptographic commitments (e.g. [8, 9]). A commitment is a kind of cryptographic digest computed on a document, or collection of documents. The commitment itself reveals no information about the document committed to, but allows any recipient of the document to verify its integrity. Testable commitments further allow the recipient of a partially redacted document to ask the document owner for proof that specific words or names have not been redacted from the document, without revealing any other information about what was actually redacted.

To illustrate how testable commitments support the verification of redacted documents, consider the simulated medical record shown in Figure 1. If the medical record is pertinent to a litigation, the requesting party sends a legal subpoena to the medical records manager, at which time the manager quickly generates a testable commitment to the record that is immediately delivered to the requestor. Next, to comply with HIPAA[1] [6], HIV medications Amprenavir and Combivir are redacted from the record and the redacted record is delivered to the requestor. If another, non-HIV-related medication is key to the opposing party's case, they may wish to verify that this medication has not been removed from the record. To do so, they can request a proof that this medication has not been redacted, and the document owner must supply it since it does not pertain to a HIPAA-protected disease. The opposing party can verify the

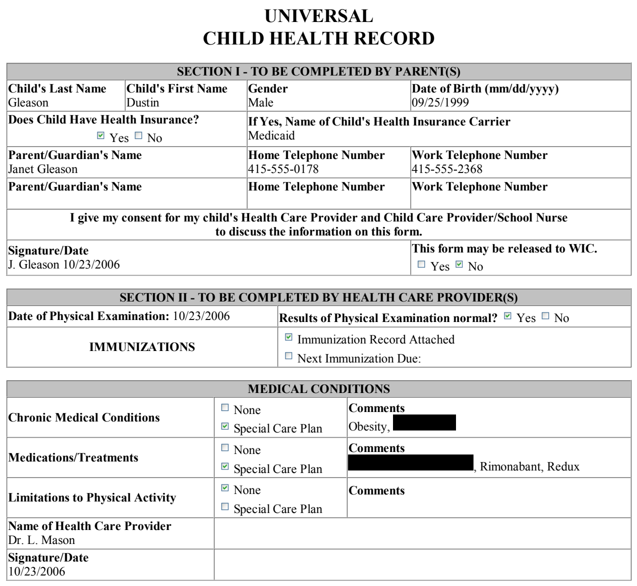
[1]HIV is one of several diseases protected by HIPAA.


UNIVERSAL
CHILD HEALTH RECORD

| | | | |
|---|---|---|---|
| **SECTION I - TO BE COMPLETED BY PARENT(S)** | | | |
| **Child's Last Name** Gleason | **Child's First Name** Dustin | **Gender** Male | **Date of Birth (mm/dd/yyyy)** 09/25/1999 |
| **Does Child Have Health Insurance?** [✓ Yes] No | | **If Yes, Name of Child's Health Insurance Carrier** Medicaid | |
| **Parent/Guardian's Name** Janet Gleason | | **Home Telephone Number** 415-555-0178 | **Work Telephone Number** 415-555-2368 |
| **Parent/Guardian's Name** | | **Home Telephone Number** | **Work Telephone Number** |
| **I give my consent for my child's Health Care Provider and Child Care Provider/School Nurse to discuss the information on this form.** | | | |
| **Signature/Date** J. Gleason 10/23/2006 | | | **This form may be released to WIC.** Yes [✓ No] |

| | |
|---|---|
| **SECTION II - TO BE COMPLETED BY HEALTH CARE PROVIDER(S)** | |
| **Date of Physical Examination:** 10/23/2006 | **Results of Physical Examination normal?** [✓ Yes] No |
| **IMMUNIZATIONS** | [✓ Immunization Record Attached] Next Immunization Due: |

| | | |
|---|---|---|
| **MEDICAL CONDITIONS** | | |
| **Chronic Medical Conditions** | None [✓ Special Care Plan] | **Comments** Obesity, HIV |
| **Medications/Treatments** | None [✓ Special Care Plan] | **Comments** Amprenavir, Combivir, Rimonabant, Redux |
| **Limitations to Physical Activity** | [✓ None] Special Care Plan | **Comments** |
| **Name of Health Care Provider** Dr. L. Mason | | |
| **Signature/Date** 10/23/2006 | | |


UNIVERSAL
CHILD HEALTH RECORD

| | | | |
|---|---|---|---|
| **SECTION I - TO BE COMPLETED BY PARENT(S)** | | | |
| **Child's Last Name** Gleason | **Child's First Name** Dustin | **Gender** Male | **Date of Birth (mm/dd/yyyy)** 09/25/1999 |
| **Does Child Have Health Insurance?** [✓ Yes] No | | **If Yes, Name of Child's Health Insurance Carrier** Medicaid | |
| **Parent/Guardian's Name** Janet Gleason | | **Home Telephone Number** 415-555-0178 | **Work Telephone Number** 415-555-2368 |
| **Parent/Guardian's Name** | | **Home Telephone Number** | **Work Telephone Number** |
| **I give my consent for my child's Health Care Provider and Child Care Provider/School Nurse to discuss the information on this form.** | | | |
| **Signature/Date** J. Gleason 10/23/2006 | | | **This form may be released to WIC.** Yes [✓ No] |

| | |
|---|---|
| **SECTION II - TO BE COMPLETED BY HEALTH CARE PROVIDER(S)** | |
| **Date of Physical Examination:** 10/23/2006 | **Results of Physical Examination normal?** [✓ Yes] No |
| **IMMUNIZATIONS** | [✓ Immunization Record Attached] Next Immunization Due: |

| | | |
|---|---|---|
| **MEDICAL CONDITIONS** | | |
| **Chronic Medical Conditions** | None [✓ Special Care Plan] | **Comments** Obesity, [redacted] |
| **Medications/Treatments** | None [✓ Special Care Plan] | **Comments** [redacted], Rimonabant, Redux |
| **Limitations to Physical Activity** | [✓ None] Special Care Plan | **Comments** |
| **Name of Health Care Provider** Dr. L. Mason | | |
| **Signature/Date** 10/23/2006 | | |

**Figure 1: The left side of the figure shows an original medical record. The right side shows the same record after the HIV medications Amprenavir and Combivir have been redacted for HIPAA compliance.**

proof against the commitment to determine that the medication was not present in the original (unredacted) medical record without learning anything more about what was actually redacted.

Testable commitments do not completely eliminate the need for a trusted third party (the judge), but they require considerably less involvement of the third party. With testable commitments, the role of the judge is limited to issuing generic rules about what terms and keywords are relevant to the case. The judge need not supervise the time-consuming application of these rules to individual documents. Testable commitments allow mutually distrustful litigating parties to apply generic rules (about which documents are responsive) to their document collections, without involving the judge or any other third party.

Testable commitments thus offer two significant advantages:

- **Efficiency.** It is much more efficient for the judge to issue generic rules about what information is responsive to the case, than to supervise the application of these rules to every document in a large collection. Testable commitments ensure that a malicious party cannot lie about whether a document contains certain responsive keywords, and thus free the judge from reviewing individual documents one by one.
- **Privacy.** With testable commitments, the judge is trusted to rule impartially on what (generic) terms and keywords are relevant to the case. The judge has no need to review individual documents, and thus need not be trusted to review the parties' confidential documents. Without testable commitments, the judge may need to review documents that are eventually found to be unresponsive. With testable commitments, the judge has no need to ever review unresponsive documents. This offers greater privacy protection to litigating parties.

We also address what may appear as limitations of testable commitments:

- **Completeness.** Testable commitments are only as good as the information committed to. A malicious party may fail to commit to its complete collection of documents, and selectively omit information known to be incriminating. The integrity guarantees offered by testable commitments are relative to the information committed to: they offer no way to audit the completeness of the commitment step. In practice, however, this may not be a severe problem. Testable commitments can be computed efficiently and reveal nothing about the data committed to. Upon receiving a subpoena, a party may thus reasonably be asked to produce a commitment to their complete collection of documents within a very short time span (e.g. within 24 hours). This short span may not leave a malicious party enough time to determine what information to exclude from the commitment. If information is excluded carelessly, or if too much information is excluded, the incompleteness of the commitment may be discovered. Unless the subpoena was expected, and the party had time to prepare a sanitized document collection, there may well be no choice but to produce a complete commitment.
- **Abusive search queries.** Testable commitments do not allow the recipient of a document to search the redacted portions of the document for arbitrary strings. Instead, the ability to test the redacted portions of the document for a particular string must be requested from the document owner. The owner may authorize some queries, and disallow others. Disagreement over what queries to allow must be resolved by a trusted third party (the judge). Note however that the involvement of the judge is limited to deciding what queries to allow. The judge need never review documents.

- **Expressiveness of search queries.** One may question the extent to which the appropriateness of redaction can be determined via simple search queries. String matching, while powerful, may be too crude a tool to interpret some natural language documents. The usefulness of testable commitments will no doubt vary by application domains.

To build a testable commitment scheme, we combine a secure commitment scheme with a trapdoor one-way function. We implemented our scheme as part of a larger tool for identifying and redacting sensitive content. In particular, the tool leverages algorithms from [12] to suggest the terms that should be tested to gain confidence that no information pertaining to a certain topic has been redacted.

**Overview.** We discuss related work in Section 2. In Section 3 we present our model and definition of testable commitments. Our construction is in Section 4. We describe an implementation in Section 5 and end with a conclusion in Section 6.

## 2. RELATED WORK

Testable commitments support the integrity verification of redacted documents, by enabling the recipient to verify that no information has been inappropriately redacted. Testable commitments are partly inspired by searchable encryption schemes that offer a similar property of controlled testing. We briefly summarize related work in the area of cryptographic commitments and searchable encryption, as well as other work on security for redacted documents.

**Commitment Schemes.** Cryptographic commitments (e.g. [5, 9]) are protocols between a document owner and a recipient in which the owner first generates a small *commitment* to the content of the document, so that any later changes will be detected because the resulting document will be inconsistent with the commitment. Testable commitments are a secure commitment scheme (as defined in Section 3) with the added ability that the recipient can request proofs concerning the value of withheld, or redacted, portions of a document. These proofs are verified against the commitment, and thus function as tests for certain values.

**Search on Encrypted Data.** Searchable encryption protocols (e.g. [11]) allow a document holder to encrypt their content in such a way that they may store it remotely on an untrusted server, but can give the server targeted *capabilities* to test for certain content values, so that when the owner needs to retrieve certain portions of their data, the server can identify the requested portions while learning nothing more. Testable commitments overlap with searchable encryption in the need for a targeted testing capability, but offer two significant additional properties. First, unlike searchable encryption, testable commitments are binding in the sense that the string committed to cannot be changed after the commitment is computed (the binding property is defined more precisely in section 3). The second important difference is that, unlike searchable encryption, testable commitments allow the receiver of a capability to verify the validity of the capability. This property is critically important in testable commitment schemes to prevent data owners from producing invalid capabilities. In searchable encryption, this property is not considered important, since there is no incentive for the data owner to generate fake capabilities (capabilities are typically used to perform search on behalf of the data owner).

**Redactable Signatures.** The notion of redactable signatures is introduced in [7]. A redactable signature allows anyone to generate a signature on a redacted document from the signature on the original (unredacted) document with no interaction with the original signer. Subsequent work has extended this basic idea to other forms of document modification, for example, sanitization [4, 1, 13]. Testable commitments offer a complementary property to redactable signatures, namely, the ability to verify redacted values in a controlled manner.

## 3. MODEL AND DEFINITION

A testable commitment scheme is an extension of a (regular) commitment scheme. We start this section with a brief review of the definition of a commitment scheme, following the terminology of [5].

### 3.1 Commitment Scheme

A commitment scheme is a two-phase protocol between a sender and a receiver. The sender's input to the protocol is a message $m$ to which the sender will commit. The receiver has no input to the protocol. In the *commit* phase, the sender computes a pair $(c, d)$ where $c$ is a commitment string and $d$ is a de-commitment string, and sends the commitment $c$ to the receiver. In a later *de-commit* phase, the sender sends the de-commitment string $d$ to the receiver. The receiver computes a $\mathsf{Receive}$ function on inputs $c$ and $d$. The function $\mathsf{Receive}$ outputs the message $m$ if $(c, d)$ is a valid commitment to $m$. Otherwise, $\mathsf{Receive}$ produces a blank output $\perp$ to indicate failure. Formally, a commitment scheme is specified by two algorithms:

- **Commit:** a randomized algorithm that takes as input a message $m$ and outputs a commitment string $c$ and a de-commitment string $d$: $\mathsf{Commit}_r(m) = (c, d)$
- **Receive:** a deterministic algorithm that takes as input a pair $(c, d)$ and outputs either a message $m$ or $\perp$: $\mathsf{Receive}(c, d) \in \{m, \perp\}$

with the following properties:

- **Soundness:** for all $m$, $\mathsf{Receive}(\mathsf{Commit}_r(m)) = m$.
- **Hiding:** the commitment $c$ reveals no information about the message $m$. More precisely, for any two messages $m_1$ and $m_2$, the functions $\mathsf{Commit}_r(m_1)$ and $\mathsf{Commit}_r(m_2)$ produce indistinguishable distributions of commitment strings. The hiding property is said to hold computationally if the distributions are computationally indistinguishable. It holds unconditionally if the distributions are identical.
- **Binding:** given a commitment $c$, it is impossible to find two de-commitment strings $d_1$ and $d_2$ such that the messages $m_1 = \mathsf{Receive}(c, d_1)$ and $m_2 = \mathsf{Receive}(c, d_2)$ are different ($m_1 \neq m_2$) and not blank ($m_1 \neq \perp$ and $m_2 \neq \perp$). The binding property may hold under some computational hardness assumption or unconditionally.

## 3.2 Testable Commitment Scheme

A testable commitment scheme adds to a regular commitment scheme the following property, which we describe first informally. The receiver can request from the sender a capability $\mathrm{Cap}(w)$ for a string $w$. This capability allows the receiver to test whether any commitment $c$ received from the sender is to message $w$. The capability reveals nothing about the message $m$ from which the commitment $c$ was computed, other than the result of the equality test between $m$ and $w$.

A testable commitment scheme consists of the following four phases:

- **Key generation.** The sender generates a public/private key pair $(pk, sk)$ and sends the public key $pk$ to the receiver.
- **Commit phase.** This phase is functionally identical to the commit phase of a regular commitment scheme: the sender sends one or multiple commitments $c$ to the receiver.
- **Test phase.** The receiver requests from the sender a capability for a string $w$. If the sender agrees to grant the request, it computes and sends the requested capability to the receiver. The receiver can use this capability to test whether any commitment received from the sender is to string $w$.
- **De-commit phase.** This phase is identical to the de-commit phase of a regular commitment scheme. The sender sends the de-commitment string $d$ to the receiver. The receiver computes $\mathsf{Receive}(c, d)$ to recover either the message $m$ or a blank output $\perp$ on failure.

Note that the test phase can be repeated any number of times for different strings, and can be interleaved in any order with commit and de-commit phases. We now describe in more detail the algorithms that make up a testable commitment scheme, and point out differences with regular commitment schemes:

- **Key generation:** a randomized algorithm that takes as input a security parameter $k$ and outputs a public/private key pair: $\mathsf{KeyGen}(1^k) = (pk, sk)$.
- **Commit:** a deterministic algorithm that takes as input a message $m$ and the secret key $sk$ and outputs a commitment string $c$ and a de-commitment string $d$: $\mathsf{Commit}_{sk}(m) = (c, d)$. Note the difference with regular commitment schemes: $\mathsf{Commit}$ takes as input the secret key $sk$ instead of a random nonce $r$.
- **Generate capability:** a deterministic algorithm that takes as input the secret key $sk$ and a string $w$ and outputs a capability for $w$: $\mathrm{Cap}(w) = \mathsf{GenCap}_{sk}(w)$.
- **Test:** a deterministic algorithm that takes as input the public key $pk$, a string $w$, a capability Cap for $w$ and a commitment $c$. If Cap is a valid capability for $w$, $\mathsf{Test}$ outputs the result of the equality test $m = w$. If Cap is invalid, $\mathsf{Test}$ produces a blank output $\perp$ to indicate failure: $\mathsf{Test}_{pk}(c, w, \mathrm{Cap}) \in \{\mathrm{Equal}, \mathrm{NotEqual}, \perp\}$.
- **Receive:** a deterministic algorithm that takes as input a pair $(c, d)$ and outputs either a message $m$ or $\perp$: $\mathsf{Receive}(c, d) \in \{m, \perp\}$

The soundness and binding properties are identical to regular commitment schemes. We define an additional soundness property for capabilities. The hiding property must be modified, to account for capabilities, and for the fact that the commitment is now a deterministic function of the message and the secret key. We define the new soundness and hiding properties as follows:

- **Capability-sound:** for all $\mathrm{Cap}(w) = \mathsf{GenCap}_{sk}(w)$ and commitment $(c, d) = \mathsf{Commit}_{sk}(m)$, we have: $\mathsf{Test}_{pk}(c, w, \mathrm{Cap}) = \mathrm{Equal}$ if and only if $m = w$ and $\mathsf{Test}_{pk}(c, w, \mathrm{Cap}) = \mathrm{NotEqual}$ if and only if $m \neq w$.
- **Hiding:** the sender runs $\mathsf{KeyGen}(1^k)$ and outputs the public key $pk$. The receiver can request any number of commitments, and any number of capabilities from the sender. The receiver then outputs two strings $m_0$ and $m_1$ for which it has not previously requested a commitment or a capability. The sender selects a random bit $b$ and outputs $\mathsf{Commit}_{sk}(m_b)$. We say that the testable commitment scheme is hiding if the receiver cannot guess $b$ with more than negligible advantage.

## 4. CONSTRUCTION

In this section, we propose our construction of a testable commitment scheme. The construction consists of a generic method for converting any commitment scheme into a testable commitment scheme, using a trapdoor one-way function.

Let $(\mathsf{Commit}^*, \mathsf{Receive}^*)$ denote a secure (hiding, binding) commitment scheme. Let $f$ denote a trapdoor one-way function (e.g. the RSA function) and let $g, h$ denote hash functions (such as SHA1 for example). In our proof, we model these hash functions as random oracles. We propose the following construction for a testable commitment:

- **Key generation:** on input $1^k$, the algorithm $\mathsf{KeyGen}$ outputs a public/private key pair $(pk, sk) = \mathsf{KeyGen}(1^k)$ for the trapdoor one-way function $f$.
- **Commit:** Given input message $m$, let $r = g(f_{sk}^{-1}(h(m)))$. We define $\mathsf{Commit}_{sk}(m) = \mathsf{Commit}_r^*(m)$.
- **Generate capability:** On input a string $w$, output $\mathrm{Cap}(w) = \mathsf{GenCap}_{sk}(w) = f_{sk}^{-1}(h(w))$.
- **Test:** On input a public key $pk$, a string $w$ and a capability Cap, check first whether $f(\mathrm{Cap}) = h(w)$. If the equality does not hold, the capability is invalid. In this case, $\mathsf{Test}$ outputs $\perp$ and halts. Otherwise, $\mathsf{Test}$ computes $c_w = \mathsf{Commit}^*_{g(\mathrm{Cap})}(w)$. If $c_w = c$, $\mathsf{Test}$ outputs Equal. If $c_w \neq c$, $\mathsf{Test}$ outputs NotEqual.
- **Receive:** $\mathsf{Receive}(c, d) = \mathsf{Receive}^*(c, d)$.

PROPOSITION 4.1. *If* $(\mathsf{Commit}^*, \mathsf{Receive}^*)$ *is a commitment scheme with the hiding and binding properties, then the testable commitment scheme defined above is sound, capability-sound, binding and hiding.*

**Proof (sketch).** It is clear that the testable commitment scheme is sound and capability-sound.
The binding property follows from the binding property of the underlying commitment scheme $(\mathsf{Commit}^*, \mathsf{Receive}^*)$. Indeed, if we can find $c$, $d_1$ and $d_2$ such that $\mathsf{Receive}(c, d_1) \neq \perp$, $\mathsf{Receive}(c, d_2) \neq \perp$ and $\mathsf{Receive}(c, d_1) \neq \mathsf{Receive}(c, d_2)$, then the same inequations hold for $\mathsf{Receive}^*$ since $\mathsf{Receive}^*(c, d) = \mathsf{Receive}(c, d)$ for all $c, d$.
We prove next that the testable commitment scheme is hiding. Let $\mathcal{A}$ be an adversary who wins the hiding game for testable commitments (defined in section 3) with non-negligible advantage. We use $\mathcal{A}$ to distinguish the distributions of commitment strings drawn from $\mathsf{Commit}_r^*(m_0)$ and

$\mathsf{Commit}^*_r(m_1)$. This contradicts our assumption that the underlying scheme $(\mathsf{Commit}^*, \mathsf{Receive}^*)$ is hiding. We answer $\mathcal{A}$'s queries as follows:

- When $\mathcal{A}$ asks for a public key, we generate a random public key $pk$ and give it to $\mathcal{A}$. Note that we do not know the associated secret key.
- When $\mathcal{A}$ asks to evaluate $g$ on input $w$, we make up a value $g_w$ and return $g_w$ to $\mathcal{A}$. We recall these values to ensure consistent replies in case $\mathcal{A}$ asks to evaluate $g$ on the same input multiple times.
- When $\mathcal{A}$ asks to evaluate $h$ on input $w$, we make up a value $x_w$, compute $h_w = f(x_w)$ and return $h_w$ for $h(w)$. As for $g$, we recall the values thus generated to ensure consistent replies.
- When $\mathcal{A}$ requests a commitment or capability for a string $w$, we either look-up the value of $h(w)$ (if we've already generated it), or generate a new value for $h(w)$ as above. Given the way $h(w)$ is generated, we know that $f_{sk}^{-1}(h(w)) = x_w$, and thus we can compute the requested commitment or capability.

After a number of queries, $\mathcal{A}$ outputs two messages $m_0$ and $m_1$. We request a commitment $\mathsf{Commit}^*(m_b)$ and pass it on to $\mathcal{A}$ as $\mathsf{Commit}(m_b)$. At this point, we distinguish two cases:

1. If $\mathcal{A}$ accepts $\mathsf{Commit}^*(m_b)$ as a valid value for the commitment $\mathsf{Commit}(m_b)$, then $\mathcal{A}$ guesses the bit $b$ with non-negligible advantage. This allows us to distinguish commitment strings drawn from the distributions $\mathsf{Commit}^*(m_0)$ and $\mathsf{Commit}^*(m_1)$, which contradicts our assumption that the underlying scheme $(\mathsf{Commit}^*, \mathsf{Receive}^*)$ is hiding.
2. If $\mathcal{A}$ can tell apart $\mathsf{Commit}^*(m_b)$ from a valid commitment $\mathsf{Commit}(m_b)$, then $\mathcal{A}$ can distinguish $g(f_{sk}^{-1}(h(w)))$ from a random value. Since we model $g$ as a random oracle, this is only possible if $\mathcal{A}$ has queried $g$ on the value $f_{sk}^{-1}(h(w))$. But if $\mathcal{A}$ has done so, we can use it as an oracle to invert $f$. This contradicts our assumption that $f$ is a one-way function.

This concludes the proof. $\square$

**Application.** Using the technique above, we can derive a testable commitment scheme from any known commitment scheme. In section 5, we describe an implementation based on the simple (heuristic) commitment scheme $\mathsf{Commit}_r(m) = h(m||r)$.

## 5. IMPLEMENTATION

A single testable commitment computed on a large file is of limited use, since it only allows the receiver to test equality between whole files. In practice, the receiver will need finer control: it must be given the ability to test equality on small textual sub-units, or blocks, of the file. This finer control can be achieved by computing separate testable commitments for all the blocks in the file (or files if the data to be committed to consists of multiple files). These blocks need not be of equal size, but rather correspond to distinct semantic entities. For example, the (short) document:

$m =$ "Currently prescribed medications are: Amprenavir, Combivir, Rimonabant"

may be divided into the following semantic blocks:

$m_1 =$ [Currently prescribed medications are]
$m_2 =$ [Amprenavir]
$m_3 =$ [Combivir]
$m_4 =$ [Rimonabant]

We do not address in this work the problem of semantic segmentation, but note that entity extraction [3] and topic segmentation tools can be used to segment documents into meaningful semantic blocks. In the example above, let us assume that the drug Accolate is key to a litigating party's case. The party may wish to confirm that Accolate has not been redacted from the record in Figure 1. With testable commitments for all blocks, the document recipient can request a proof that the redacted term is not Accolate. The recipient can verify the proof against the commitment received earlier to determine that the term in question has not been redacted, while learning nothing more about what has been redacted.

**Communication and storage efficiency.** If a testable commitment is computed for every small block in a large file, the owner of the file may need to send a large number of commitments to the receiver. To improve the communication efficiency of the commitment step, we discuss next how multiple testable commitments may be combined into a single committed value with a Merkle Hash Tree.

### 5.1 Merkle Hash Tree

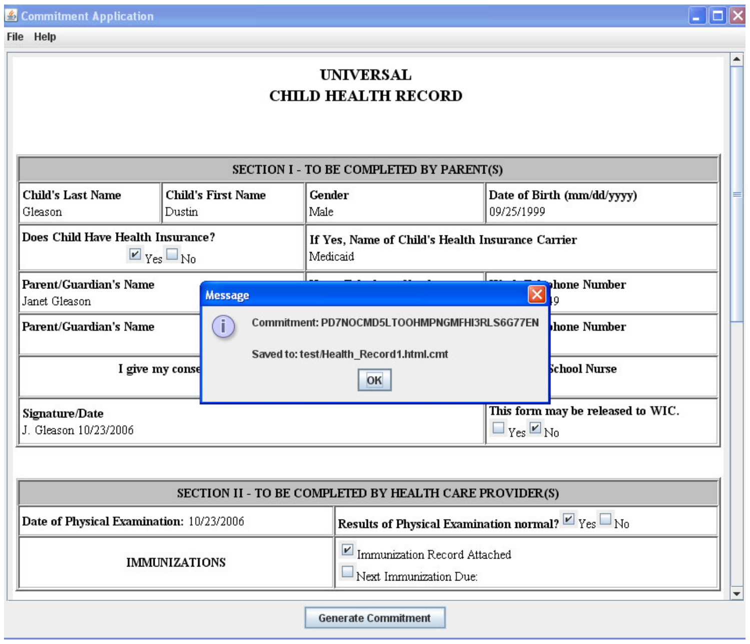
Let $h$ denote a hash function, and let $c_1, \ldots, c_n$ denote a collection of commitments. In what follows, we assume for simplicity that $n = 2^k$ for some $k$. A Merkle hash tree [8] computed on this collection is a binary tree whose nodes are tagged as follows. The leaves of the tree are tagged with the values $h(c_1), \ldots, h(c_n)$. If $N$ is an interior node of the tree, let $L$ denote the tag associated with the left child of $N$, and $R$ denote the tag associated with the right child of $N$. We associate with $N$ the tag $h(L||R)$. The root of the tree serves as a commitment to the collection $c_1, \ldots, c_n$.

**Authentication path.** Let $B$ be a leaf of a Merkle hash tree $T$. We call an authentication path for $B$ the set of siblings of all the nodes on the path from $B$ to the root of $T$. Given the root of $T$, the authentication path proves membership of $B$ in the tree $T$ and can be used to implement $\mathsf{Receive}$.

### 5.2 Implementation

We built a prototype testable commitment system based on the construction described in section 4. Given a document, the system produces a testable commitment of that document. The software first removes common English "stop" words (e.g. "and", "the", etc.) and produces a Merkle Hash Tree (using SHA-1) on the remaining terms. We used 1024-bit RSA as the one-way function. Computing a commitment was the most performance-intensive step of the protocol, as it involved one private-key operation per term.

We used RFC 2246 [2] as a test document. This RFC contained 13092 terms after eliminating common English stop words. Our prototype took 135 seconds to produce a commitment on a standard desktop (3.0 GHz Pentium D with 3.5 GB RAM). This is roughly 97 terms per second. Most of the time was taken by the private key encryption operation



UNIVERSAL
CHILD HEALTH RECORD

| SECTION I - TO BE COMPLETED BY PARENT(S) | | | |
|---|---|---|---|
| Child's Last Name<br>Gleason | Child's First Name<br>Dustin | Gender<br>Male | Date of Birth (mm/dd/yyyy)<br>09/25/1999 |
| Does Child Have Health Insurance?<br>[✓ Yes] No | | If Yes, Name of Child's Health Insurance Carrier<br>Medicaid | |
| Parent/Guardian's Name<br>Janet Gleason | | | hone Number<br>9 |
| Parent/Guardian's Name | | | hone Number |
| I give my conse | | | School Nurse |
| Signature/Date<br>J. Gleason 10/23/2006 | | | This form may be released to WIC.<br>Yes [✓ No] |

Message

Commitment: PD7NOCMD5LTOOHMPNGMFHI3RLS6G77EN

Saved to: test/Health_Record1.html.cmt

| SECTION II - TO BE COMPLETED BY HEALTH CARE PROVIDER(S) | |
|---|---|
| Date of Physical Examination: 10/23/2006 | Results of Physical Examination normal? [✓ Yes] No |
| IMMUNIZATIONS | [✓ Immunization Record Attached]<br>Next Immunization Due: |

**Figure 2: Screen shot of the application creating a testable commitment**

for each term. Computing a capability and verifying a capability were relatively fast, each taking roughly a hundredth of a second.

Caching the private-key encryptions speeds things up considerably. For instance, of the 13092 terms in RFC 2246, 10850 are repeats. Using caching, our prototype took roughly 23 seconds to generate a commitment for RFC 2246.

In Figure 2 we show the application generating a testable commitment for the medical record of Figure 1.

## 6. CONCLUSION

We propose a commitment scheme with new functionality: our testable commitment scheme allows a sender to commit to a document, then prove to a receiver, after redacting parts of the document, that the redacted portions of the document do not contain some specific keywords. This functionality would be useful in particular in litigation cases. Testable commitments can help litigating parties demonstrate in a privacy preserving way that the redaction applied to a document does not hide any data responsive to the case. We present a construction and implementation of a testable commitment scheme. Our implementation shows that testable commitments are efficient enough for practical deployment in applications involving medium-sized documents.

## 7. REFERENCES

[1] G. Ateniese, D. Chou, B. de Medeiros and G. Tsudik. Sanitizable signatures. In *ESORICS 2005*.

[2] T. Dierks and C. Allen. The TLS Protocol, IETF RFC2246, January 1999.

[3] GATE: General Architecture for Text Engineering. On the web at `http://gate.ac.uk/projects.html`

[4] S. Haber, Y. Hatano, Y. Honda, W. Horne, K. Miyazaki, T. Sander, S. Tezoku and D. Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In *ASIACCS*, 2008.

[5] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Proc. of Crypto '96*, pp. 201–215.

[6] Health Insurance Portability and Accountability Act of 1996. Public Law 104-191, 104th Congress. `http://www.cms.hhs.gov/HIPAAGenInfo/Downloads/HIPAALaw.pdf`

[7] R. Johnson, D. Molnar, D. Song and D. Wagner. Homomorphic signature schemes. In *CT-RSA 2002*.

[8] R. Merkle. A digital signature based on a conventional encryption function. In *Crypto '87*, pp. 369–378.

[9] M. Naor. Bit commitment using pseudorandomness. In *Journal of Cryptology*, 1991.

[10] Pennsylvania Superior Court, Appeal form the order of January 31, 2000. Filed February 8, 2001. Appellant, George A. Spisak, Jr., Appellee, Margolis Edelstein.

[11] D. Song, D. Wagner and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Security and Privacy Symposium*, May 2000.

[12] J. Staddon, P. Golle and B. Zimny. Web-based inference detection. In *USENIX Security 2007*.

[13] R. Steinfeld, L. Bull and Y. Zheng. Content extraction signatures. In *ICISC* 2001.