Matchbox: Secure Data Sharing

Homeland security requires that organizations share sensitive data, but both suppliers and users must typically restrict data access for security, legal, or business reasons. Matchbox database servers provide highly secure, fine-grained access control using digitally cosigned contracts to enforce sharing restrictions. To handle security operations, Matchbox uses the tamper-responding, programmable IBM 4758 cryptographic coprocessor. Matchbox servers can be distributed on a network for high availability, and parties can communicate with Matchbox over public networks — including hostile environments with untrusted hardware, software, and administrators.

Providing database security raises many challenges, particularly for homeland security applications, where users desire Internet access to highly sensitive data. This easy access must be balanced against restrictions on sharing due to security, legal, and business concerns. Example applications here include:

- Government agency watch lists, which become useless if security is breached and the information is made public.
- Healthcare industry data, which, when aggregated, can indicate disease patterns, health risks, or even insurance fraud. Laws restrict data sharing in this area to protect patient privacy.
- Bank transaction records, which can indicate money laundering, identity theft, or fraud. However, banks resist sharing customer data, and don’t trust third-party aggregators.

To protect sensitive data, a server must let users confidentially query a database and access data without viewing unauthorized content. It must let users receive results from other user queries. It must protect against false alarms, such as when an attacker sends a bogus alert. Given the need for high availability, the database must be distributed without making the server vulnerable to attack. Finally, the server must contend with questions of trust: How can users be sure the database is current if the system administrator is untrusted? How can the server prove to users that the database hardware and software are trustworthy when users are communicating with a remote server over an untrusted network?

Existing commercial database products focus on information providers’ concerns, offering features that restrict remote user access and let database administrators...
To provide Matchbox-like security features for database systems, developers can use the Trusted Computing Group (TCG; www.trustedcomputinggroup.org) architecture or private information retrieval (PIR). However, these technologies cannot offer the same security guarantees as Matchbox provides.

**Trusted Computing Group Architecture**

The TCG architecture uses a trusted platform module (TPM) as a root of trust. Bootstrap, operating-system, and application code build upon the root to let users trust a platform.1 However, TCG cannot itself provide Matchbox-type security guarantees for two reasons.

First, the TPM has minimal physical security. It assumes that a user physically present at the platform will not mount a sophisticated attack. In contrast, the IBM 4758 assumes that the user is an adversary.

Second, the TPM has a fixed command set and relies on its host to run the application. Queries and database data will be in the clear on the host, subject to attack. Matchbox assumes that the server is in a remote, hostile setting, and keeps all clear text data within the coprocessor’s secure boundary.

However, TCG is an excellent complement to Matchbox, which doesn’t secure client data that must be in clear text, and doesn’t address storage of client-party private keys. A TPM-based client can secure private keys and require authorization for their use. It also ensures that keys are not released unless the client software stack, from the OS to the application, is in a known certified state, free from viruses and Trojans.

**Private Information Retrieval**

PIR is a set of software techniques that hide queries and results from a server (see www.cs.bgu.ac.il/~beimel/Research/PIR.html for an overview). To accomplish this, PIR uses various techniques, including reading the entire database, reading random indices, querying through a trusted anonymizer, and using multiple servers to process a request.

Query hiding is a Matchbox goal, and the server uses PIR techniques such as blinding writes (which hide the identity of the database record being written), and keyed hashing of database indices, as well as padding responses and sending decoys to deter traffic analysis.2

Matchbox also hides queries in a more fundamental way. PIR addresses the case in which remote queries to a central database must be hidden, and the database owner is the adversary. Matchbox posits that the database server is remote to both the requestor and the information provider. In Matchbox, those parties, and the server administrator, are all considered potential attackers. Therefore, the host server does not have access to any data, whether it be queries, updates, responses, or database tables.

**Database Security**

Typical commercial database products provide security features such as

- table encryption,
- views (to limit access),
- access control for authorized users, and
- user and password authorization.

These features can restrict a remote user’s access to a database, table, or even a row, and define authorized operations and how results can be used.

**Matchbox Advantages**

Unlike most commercial database products, which focus on the information providers’ concerns, Matchbox provides a more comprehensive solution that offers three key security guarantees:

- *Information providers can distribute databases over a public network while keeping contents private and controlling content access.* Commercial databases assume a fully trusted adminis-
Homeland Security

The Administrator Role

No administrator, regardless of software privilege or physical access to a server, can violate the Matchbox security guarantees. With standard, software-only solutions, a compromised administrator can — through hardware or software means — add, delete, or alter data; restore outdated contents; view or intercept queries or notifications; or even replace all or part of the operating system or program. With Matchbox, the only attack available to administrators is denial of service. The Matchbox administrator creates user accounts on a server and loads configuration data on behalf of the users; he or she can thus cause a DoS by deleting or refusing to load data. Also, the host server administrator can cause a DoS by interfering with the network or by altering or deleting disk files. Although it cannot prevent such attacks, Matchbox guarantees that they will be detected.

Matchbox Components

Matchbox combines a trusted program and a secure hardware coprocessor with software contracts agreed to by all parties.

Coproessor

The Matchbox server program runs on an IBM 4758 programmable cryptographic coprocessor (see www-3.ibm.com/security/cryptocards/html/overproduct.shtml for complete details). The coprocessor’s sealed module incorporates physical-penetration, power-sequencing, temperature, and radiation sensors to detect physical attacks against the encapsulated subsystem (see Figure 1). Batteries provide backup power, which is active from factory certification through the end of the product’s useful life. Any detected tamper event results in loss of power. In such cases, the coprocessor imme-
diately deletes internal secrets, such as cryptographic keys, and destroys its factory certification.

The US National Institute of Standards and Technology certified the IBM 4758 at Federal Information Processing Standard (FIPS) 140-1 level 4 — its highest certification and the benchmark standard for evaluating commercial cryptographic products’ security and algorithmic implementations (see www.itl.nist.gov/fipspubs/fip140-1.htm).

The Matchbox server receives user requests, accesses the database, and sends notifications. It triple-DES encrypts network communications and database contents, uses message authentication codes (MAC) to protect content integrity, and is secured against replays. Requests, notifications, and the server’s database contents exist in clear text only within the secure confines of the IBM 4758.

Contracts and Tables
Each request to a Matchbox server invokes a contract — a script that is digitally cosigned by the database table owner, database table users (the parties making processing requests), and third parties who might receive a notification (the result) based on the user request.

A contract lists the processing steps Matchbox will take in response to a user request. By signing the contract, the parties agree that

• only cosigning requestors can execute the contract,
• access is permitted only to a specific database table owned by a cosigner, and
• notification responses can be sent only to a cosigner.

A contract step defines an action, such as adding or deleting a table item, querying a table, performing a limited operation on the query data, receiving specified query results, and sending a notification.

Database table owners sign each of their tables, establishing them as table owners and securely defining the table columns and search keys.

Example Problem
To show how Matchbox works, we’ll use the example of an airline screening passengers against government agency watch lists. Figure 2 shows the key parties — the watch-list-issuing agencies, law enforcement officials, and the airline.

Security Issues
As in most homeland security applications, the security issues in this scenario are complex. The airline must have access to the database, but if watch lists become public, they’re worthless. Location is also a concern. Placing the watch list at a contributing agency is an incomplete solution; performance and availability concerns make a distributed, onsite, or outsourced database more practical, as does the fact that the airline might not want the agency to see its passenger lists. But placing the watch list at the airport has its own problems:

• Being outside the agency’s control subjects the server to physical attack.
• The agency must be able to securely update the database.
• The airline must be sure that it’s accessing the most recent data (a system administrator might attempt to attack the system by replacing the watch list with a previous backup).

Also, after screening, the agency might not want the results to flow back to the airline: an untrusted reservation agent should not know particular names on a watch list. Instead, the airline should simply receive a confirmation that the screening occurred, while the actual results go to a third party, such as a
law enforcement office. Finally, other third parties—such as auditing and billing operations—should receive notifications of screening events. The data they receive should be restricted, and the notifications themselves must be invulnerable to diversion and impersonation.

**Contract Example**
Matchbox enforces access control at each contract-processing step. Requests are authorized, database access granted, and notifications sent only if the contract has been signed by the requestor, the table owner, and the third party, respectively. In our airline screening example, a watch-list table might contain a first and last name, a passport country and number, and an alert severity. As Figure 2 shows, a typical screening might feature the following steps:

1. Receive a request from an airline containing a picture of the passenger, and his or her passport country and number.
2. Read a database table row based on the passport country and number.
3. Perform a matching operation to determine whether the passport is on a watch list.
4. If there is a match and the severity is high, send a notification to law enforcement containing the passenger’s photograph and first and last name. Otherwise, send a decoy so that a network observer cannot detect a match through traffic analysis.
5. If the law enforcement destination is offline, use an alternate.
6. Send a notification to an auditor containing the requestor and the passport country and number.
7. Send a notification to billing containing the requestor.
8. Send a response to the requestor. For a high-severity match (such as an individual on a classified watch list), the response might indicate only that the request was processed, while hiding the actual results. For a lower severity match (such as an individual with a tendency to get drunk on flights) the response might indicate that there is a match and include the actual alert contents.

A typical contract in this case might specify the following:

- Each request will be handled only pending agreement of the airline, the government agencies, the law enforcement agencies, and auditing and billing services.
- If there is a high-severity match, law enforcement will see the passenger’s photograph and first and last name.
- The auditor sees the requestor and request, but not the match results.
- The billing agency sees only the requestor.
- Use decoys to prevent observers from detecting a high-severity match.
- Only law enforcement and the auditor will see the request.
- As always, all information is hidden from physical or software attack at the server or on the network.

Matchbox contains no preprogrammed contracts or tables—parties can define, cosign, and download contracts and tables to Matchbox after deployment without interrupting server operation.

**Four Barriers to Attack**
Matchbox servers combine four features to achieve its security guarantees: request processing, database security, network security, and server attestation.

**Request Processing**
The Matchbox server is basically a request processor. Each request is issued against a contract. Matchbox enforces security at each step. First, as part of server configuration, the administrator loads a table configuration. Matchbox verifies the signature of the table configuration, establishing an owner and preventing alteration. The second step in server configuration is contract loading. Because the parties cosign it, the contract forms an agreement between all parties rather than relying on a trusted third party or a public-key certificate authority.

Third, when a party issues a request against a contract, Matchbox checks the requestor against the contract to ensure that the requestor is authorized to execute the contract, and that the contract has not been replaced, tricking the requestor into executing the wrong contract script. This check protects both the server and requestor.

Fourth, because the contract specifies table access, Matchbox checks to ensure that the table owner has signed the contract. This ensures that the requestor cannot access a table unless the owner has granted permission, and that the requestor is accessing an authentic table rather
than a bogus replacement. This check protects both the table owner and the requestor.

Finally, the contract might specify one or more notifications and their content. After processing a request, Matchbox checks that the notification party has signed the contract to ensure that a notification cannot be blocked without detection, and that the requestor cannot send a notification unless the destination party has agreed to it. This check protects both the requestor and the notification party.

In addition to these standard request-processing steps, Matchbox offers several optional services to improve contract processing security.

- **Contracts can include conditional steps.** For example, if a notification fails, the server can route the notification to another third party rather than just failing. In our screening example, if one law enforcement destination is down, the contract can indicate an alternate.

- **Matchbox can pad a contract-processing step or group of steps to constant time.** This option prevents timing attacks when the processing time might leak information. In the screening example, a fast response might indicate a watch list match, and a slow response could indicate no match. Time padding the matching step would keep an observer from gaining this knowledge.

- **Contracts can specify decoy notifications and notification padding to avoid leaking information about the data size.** In the screening example, a match triggers a notification that includes the individual’s picture among other data. If there is no match, the contract specifies a decoy message to keep observers from gaining information based on the notification’s presence or absence. Further, the contract could specify that all notifications be padded to constant length to avoid information leakage based on notification length.

Finally, Matchbox can use blinding table updates to mask database contents. In the screening example, an attacker should not be able to tell that a particular table row is empty by observing the pattern of updates. Matchbox can optionally blind the updates by writing some or all rows — either on all updates or the initial insert — so that observers can’t determine whether a row has data.

**Database Security**

Database tables are stored on the untrusted host system disk, and Matchbox assumes that the host is completely vulnerable to attack. Because it also assumes that the commercial host database software is untrusted, Matchbox doesn’t depend on the database software features for security.

Matchbox employs a unique set of cryptographic keys for each database table. To accommodate a large number of tables, the table keys are themselves stored in the database, protected by a master key. The master key is generated on the coprocessor, stored in battery-backed RAM, and never leaves the card. Any attempt to probe the card erases the master key.

In addition, Matchbox includes three other security features:

- **Encryption for privacy.** All data stored on the host disk is triple-DES encrypted. A nonce (a nonrepeating value) is added to each clear text database row. This ensures that an observer cannot determine from the cipher text that two clear text rows are identical when encrypted using the same cryptographic key.

- **MAC for integrity.** Each row includes a MAC (a keyed hash) of the cipher text to prevent data alteration. The MAC includes the table number and row, so an attacker cannot exchange rows within or between tables.

- **Versioning to prevent a database table replay.** Each table row contains a version number to prevent a database table replay. That is, an attacker cannot substitute a previous database version to undo a table update. The version numbers for all rows in the table are themselves stored hierarchically in the table (encrypted, MAC’ed, and versioned to prevent attack). The final (root) version number is stored in the coprocessor’s battery-backed RAM. This value is incremented on each update, and can never be reset.

Matchbox also offers two optional services to improve database security. First, it can pad rows to constant length to avoid leaking information about the data. Second, it can hide the row position by hashing the search key with a hash key. This prevents information leakage based on row access patterns.

**Network Security**

Contracting parties communicate with the Matchbox server over the Internet, subjecting their communications to attack through observation or alteration. As with the database, Matchbox protects network data using triple-DES encryption, including a nonce; a MAC to prevent data alteration; a sequence num-

---

IEEE INTERNET COMPUTING www.computer.org/internet/ NOVEMBER • DECEMBER 2004
The encryption and MAC network session keys are exchanged using a protocol based on the versatile Secure Key Exchange Mechanism for the Internet (SKEME), which forms the basis of some IPsec modes. The administrator simply loads the party’s networking public key into Matchbox. However, this doesn’t give the party any authority regarding contract or table access; such authority requires the party’s signing key. Only the party (not the administrator), can load its signing public key, because it requires a challenge/response. This proves to Matchbox that the party holds the private key related to the signing public key, and is the basis for granting the party access to contracts and tables.

**Attestation**

The entire Matchbox security effort would be for naught if an attacker could simply replace Matchbox with a clever software impersonation. Matchbox is intended for installation in insecure locations, where physical security is impractical and the end user or administrator might also be the adversary. Given this, contracting parties must be assured that they’re talking to an authentic Matchbox server.

Matchbox leverages the IBM 4758’s attestation mechanism to assure remote parties that they’re talking to an untampered IBM 4758 coprocessor. This mechanism can attest that the programmable coprocessor is running a known, trusted version of the Matchbox code running under a known, trusted version of the OS, and loaded by a known, trusted version of the bootstrap code. It does this as follows:

- At the factory, the card generates the device key pair and IBM signs a certificate attesting to the device public key and loads it on the card.
- When the bootstrap code is loaded, the card generates a bootstrap key pair. The device key signs a certificate attesting to the bootstrap public key and a hash of the bootstrap code.
- When the Matchbox program is loaded, the card generates an OS key pair. The bootstrap key signs a certificate attesting to the OS public key and a hash of the OS and Matchbox code.
- When the Matchbox program is first run, it generates its networking-communication key pair. The OS key signs a certificate attesting to the networking public key.

When a party connects to a potential Matchbox server, it asks for its networking public key. The party receives a certificate chain back to the IBM 4758 manufacturing process root public key. By verifying the certificate chain, the party can attest that this public key belongs to a true IBM 4758 running as a Matchbox server.

Each time a party establishes a secure network connection to the server, it issues a challenge based on the server’s public key. Once the server answers the challenge, it proves to the client party that it has the private key associated with the public key. Because the IBM 4758 will never reveal its private key, and the coprocessor responds to tampering by erasing the key, the party can be sure that the server is authentic.

**Conclusion**

We designed Matchbox software to be extensible, so that the contract process can add, register, and update a database with new functions. Current functions include those for configuring the system, creating and updating a database, performing matches on database fields, and sending notifications. Functions for other applications (such as aggregation of financial or medical data) can be developed as applications emerge.

Matchbox currently addresses server security, and the demonstration applications store client authentication keys with only password protection. A more secure system should address client security as well, with keys protected by biometric data or a hardware security module such as a smart card or a trusted platform module.

**References**


**Kenneth Goldman** is an advisory engineer at the IBM T.J. Watson Research Center in Hawthorne, New York. His interests include secure applications and hardware security. He has an MSEE from Massachusetts Institute of Technology. Contact him at kgold@watson.ibm.com.

**Enriquillo Valdez** is a researcher at IBM’s T.J. Watson Research Center. His technical interests include security, secure coprocessors, and hypervisors. He received a PhD in computer science from Polytechnic University. Contact him at rvaldez@us.ibm.com.