Tamper-Proof Privacy Auditing for Artificial Intelligence Systems

Andrew Sutton and **Reza Samavi**
McMaster University, Computing and Software, Ontario, Canada
suttonad@mcmaster.ca, samavir@mcmaster.ca

Abstract

Privacy audit logs are used to capture the actions of participants in a data sharing environment in order for auditors to check compliance with privacy policies. However, collusion may occur between the auditors and participants to obfuscate actions that should be recorded in the audit logs. In this paper, we propose a Linked Data based method of utilizing blockchain technology to create tamper-proof audit logs that provide proof of log manipulation and non-repudiation.

1 Introduction

Health data analytics and Artificial Intelligence (AI) enabled diagnostics (e.g., IBM Watson cognitive computing) are increasingly becoming part of healthcare processes. Carefully designed and continuously updated AI systems rely on the collaboration between diverse researchers (e.g., medical scientists, computer scientists, statisticians) and the processing of immense private data to provide prediction and diagnosis of conditions (e.g., health research). Protecting the privacy of individuals who contribute their data to such a collaborative research environment is a challenging task, particularly when individual's personal information is passed between different organizations, researchers, and AI systems that might operate under different jurisdictions.

Data sharing agreements (DSA) are legally binding documents established between organizations that detail the policies and conditions related to the sharing of personal data [Swarup *et al.*, 2006]. Auditing the use and transformation of the data is essential to enforcing accountability, especially when AI systems are part of the collaborative process. The scenario in Fig. 1 demonstrates a collaborative research environment where the research teams must comply with the DSAs and are monitored for their compliance through the use of privacy audit logs. Auditors are responsible for checking compliance with the DSA by examining the privacy logs generated by the research teams [Samavi and Consens, 2018].

In the scenario in Fig. 1, there is a problem in the trust placed in an auditor and the audit log itself. If the auditor works for the organization that they are auditing then the quality of the audit depends on influencing factors between the organization and the auditor [Anderson, 2016]. Collusion

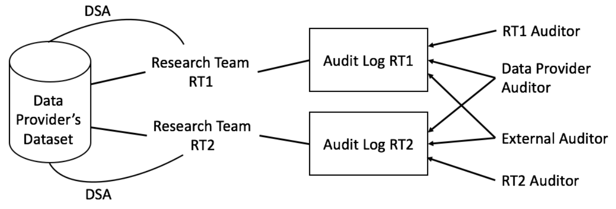


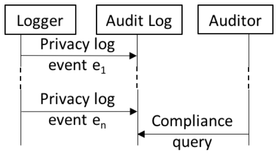
Figure 1: Example auditing scenario in a data sharing environment

can occur between individuals in the organization, such as researchers in the research teams, and the auditor to obfuscate or modify the integrity of the generated logs. Inadvertent breaches of privacy policies by AI systems must also be captured in the generated logs and should not be tampered with. The resulting degraded trust placed in the auditing process is a problem that needs to be solved in order to prove that organizations are responsible for privacy breaches resulting from non-compliant actions or to prove that they are compliant with the policies. In order to combat the potential modification of the logs due to collusion between auditors and researchers or inadvertent breaches of data usage by AI systems, a mechanism to provide tamper-proof audit logs is needed [Spoke, 2015].

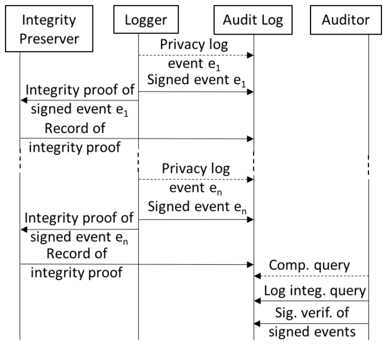
In this paper, we propose a Linked Data-based [Heath and Bizer, 2011] model for creating tamper-proof privacy audit logs and provide a mechanism for log integrity and authenticity verification that auditors can execute in conjunction with performing compliance checking queries. Section 2 presents how privacy audit logs are generated and the design requirements of our model. Our solution to generate tamper-proof privacy audit logs is described in Sect. 3. In Sect. 4, the results of an experiment to validate the scalability of our method is given. Section 5 provides an investigation of the related work. Concluding remarks are discussed in Sect. 6.

2 Properties of Tamper-Proof Privacy Logs

Privacy auditing addresses three characteristics of information accountability: validation, attribution, and evidence [Weitzner *et al.*, 2008]. Validation verifies a posteriori if a participant has performed the tasks as expected, whereas attribution and evidence deal with finding the responsible participants for non-compliant actions and producing supporting evidence, respectively [Weitzner *et al.*, 2008]. To address



(a) Privacy audit log generation process


Integrity Preserver
Logger
Audit Log
Auditor
Privacy log event e1
Signed event e1
Integrity proof of signed event e1
Record of integrity proof
Privacy log event en
Signed event en
Integrity proof of signed event en
Record of integrity proof
Comp. query
Log integ. query
Sig. verif. of signed events

(b) Tamper-proof log generation process

Figure 2: Privacy audit log generation comparison

these characteristics, privacy audit logs need to capture events with deontic modalities, such as capturing privacy policies, purpose of data usage, obligations of parties, and data access activities. A privacy audit log generation process is depicted in Fig. 2a. The process is composed of a logger producing log events of promised and performed privacy acts and storing them in an audit log accessible to auditors. The logger generates multiple privacy log events (e_1 to e_n) over time (e.g., expressing privacy policies, requesting access and access activities). An auditor can then perform compliance queries against the audit log to determine if the performed acts are in compliance with the policies in the governing DSA (e.g., the scenario in Fig. 1) [Samavi and Consens, 2014].

In this research, we utilize the L2TAP privacy audit log because it provides an infrastructure to capture all relevant privacy events and provides SPARQL solutions for major privacy processes such as obligation derivation and compliance checking [Samavi and Consens, 2018]. The L2TAP model follows the principles of Linked Data to publish the logs. By leveraging a Linked Data infrastructure and expressing the contents of the logs using dereferenceable URIs, the L2TAP audit log supports extensibility and flexibility in a web-scale environment [Samavi and Consens, 2018].

An event in a privacy audit log needs to be non-repudiable so that the performed act cannot be denied and the authenticity of the event can be provably demonstrated. For example, in the scenario in Fig. 1, if an auditor determines that the researchers have performed non-compliant actions, there is no provable method of holding the researchers accountable for their performed acts. Furthermore, after being logged, log events should not be altered by any participant, including the logger and auditor. If the researchers and auditors act in collusion to hide non-compliant acts in the log to avoid consequential actions, the resulting log does not represent the true events. Without a mechanism to provably demonstrate that the integrity of the log is intact, there will be a significant lack of confidence in the auditing process [Spoke, 2015]. The privacy audit log should enable the logger to digitally sign an event to support non-repudiation and offer a mechanism to preserve the integrity of log events (e.g., hashing or encryption). Verifying the signature of an event will prove the authenticity of the event logger and the ability to verify the integrity of the log events will result in a genuine audit of the participant's actions, since the performed actions (events) in the log are proven to be authentic.

Figure 2b depicts the additional steps required in the privacy audit log generation process to support event non-repudiation and integrity. The log is generated by the logger, but an additional entity, the integrity preserver, is required. After a log event is generated, the event must be signed by the logger to support provable accountability. Integrity proof digests (i.e. cryptographic hashes) of the log events should be generated and stored by the integrity preserver as the immutable record of the integrity proof. These records can then be retrieved to enhance the process of compliance checking with log integrity verification.

3 Blockchain Enabled Privacy Audit Logs

In situations where a central authority has control over information resources, the trust placed in that authority to maintain correct and accurate information is reduced because there is no provable mechanism for external entities to verify the state of the resources. Blockchain technology solves the trust problem by maintaining records and transactions of information resources through a distributed network, rather than a central authority [Kehoe *et al.*, 2015]. The use of blockchain technology to create an immutable record of transactions is analogous to the auditing problem we are trying to solve; the need for the immutable storage of information that is not governed by a central authority. In this section, we present how our blockchain enabled privacy audit log model works. We start with a brief background on the blockchain technology leveraged by our model, the Bitcoin blockchain, in Sect. 3.1. We describe the architecture of our model in Sect. 3.2 and the log integrity verification process in Sect. 3.3.

3.1 Bitcoin Blockchain

The Bitcoin system [Nakamoto, 2008] is a cryptocurrency scheme based on a decentralized and distributed consensus network. Transactions propagate through the Bitcoin peer-to-peer network in order to be verified and stored in a blockchain. A blockchain is a decentralized database comprised of a continuously increasing amount of records, or blocks, that represents an immutable digital ledger of transactions [Pilkington, 2016]. Each block in the blockchain is composed of a header containing a hash of the previous block in the chain (forming a chain of blocks) and transaction payload. Transactions are written to the blockchain through data structures that contain an input(s) and output(s). The Bitcoin blockchain allows up to 80 bytes of additional storage to a transaction output using the OP_RETURN opcode available through Bitcoin's transaction scripting language [Cucurull and Puiggalí, 2016]. A blockchain explorer application programming interface (API) and explorer web application are required to query transaction and block information on the Bitcoin network. We utilize this queryable special transaction to store an integrity proof of privacy audit logs on the Bitcoin blockchain.

3.2 Architectural Components

A blockchain is well suited to fill the role of the integrity preserver in the tamper-proof log generation process in Fig. 2b.

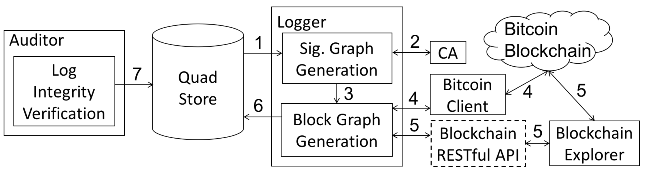


Figure 3: The architecture of the model

We use the capabilities provided by the Bitcoin blockchain to store an immutable record of the log integrity proofs. The logger generates privacy log events and signs these events. After producing integrity proofs of the signed events, each of the proofs will be written to the Bitcoin blockchain through a series of transactions. The components for signing log events and creating Bitcoin transactions are *signature graph generation* and *block graph generation* illustrated in Fig. 3 and described below.

The signature graph generation component is responsible for capturing the missing non-repudiation property of the L2TAP audit log framework. An L2TAP audit log is composed of various privacy events such as data access requests and responses. The log events consist of a header that captures the provenance of an event and a body containing information about the event, such as what data is being accessed by whom. URIs are used to identify a set of statements in the header and body to form RDF named graphs stored in a quad store [Carroll *et al.*, 2005]. We generate a new named graph, called the *signature graph*, that contains assertions about the event's signature, including the signature value and how to verify the signature. The event that will be signed is pulled from the quad store and signed by the logger (flow 1). The signature graph generation follows the process in [Kasten *et al.*, 2014; Kleedorfer *et al.*, 2016], which includes the graph digest algorithm in [Sayers and Karp, 2004] to compute the digest of the event graph. The logger signs the event digest with their private key. There needs to be a public key infrastructure (PKI) with certificate authorities (CA) in place where the logger has a generated key pair used for digital signatures (flow 2). The computed signature and signature graph will be passed to the block graph generation component to be part of the integrity proof digest computation (flow 3).

The block graph generation component conducts transactions on the Bitcoin network to write the integrity proof digest to the blockchain. Formally, an integrity proof is a cryptographic hash of an event graph and corresponding signature graph, where $integrityProof = \Pi_{i=0}^{n} h(s_i) mod(p)$, n is the number of statements in the graphs, h is a cryptographic hash function (e.g., SHA-256), s_i is a graph statement, and p is a large prime number. The logger uses a Bitcoin client to create a transaction containing the integrity proof digest (flow 4) to be written to the blockchain. After the transaction is written to the blockchain, the transaction data is queried through a RESTful request [Rodriguez, 2008] to a blockchain explorer API (flow 5). The queried data is parsed to an RDF named graph, called the *block graph*. The block graph contains the integrity proof digest and information identifying the block containing the transaction on the blockchain. After the block graph has been generated, it is stored in a quad store in order for an auditor to perform log event integrity and signature verification queries (flows 6 and 7, respectively). Generating a block graph reduces the burden on the auditor when performing log integrity verification since all of the event integrity proofs are stored in a quad store. Without the block graphs, the auditor would have to search the *entire* Bitcoin blockchain for the integrity proof digests. Since the Bitcoin blockchain is a public ledger, there are many transactions unrelated to the auditor's search, which would make this method of searching inefficient. An alternative approach is to use a full Bitcoin client to download the entire blockchain, however in this case the required network bandwidth and local computing power are major limitations.

3.3 Log Integrity Verification

The goal of an auditor in a privacy auditing scenario is to check the compliance of participants' actions with respect to the privacy policies. The authors in [Samavi and Consens, 2014] described a SPARQL-based solution for compliance checking (i.e., determining if data holders have followed the access policies for given data access activities). We extended the SPARQL-based solution to enhance the compliance checking queries described in [Samavi and Consens, 2014] to include the integrity and authenticity verification of log events.

For a given L2TAP log, the process of verifying the log integrity and authenticity and compliance checking can be performed in a sequence; i.e., for all events in the log, first ensure the integrity and authenticity of all events and then execute the compliance queries for the interested access request. However, in practice this approach is not desirable as for a fast growing log, verifying the entire log for each audit query is very expensive (see our experiment in Sect. 4). Alternatively, we can devise an algorithm that verifies the integrity and authenticity of a small subset of the event graphs for a given access request. The L2TAP ontology provides compliance checking of a subset of events through SPARQL queries [Samavi and Consens, 2018], which our integrity verification procedure can leverage to reduce the runtime.

Prior to an auditor checking compliance, they must verify the integrity of a log event and the event signature. The interested event and signature graphs are queried from the quad store. Since a cryptographically secure hash function was used to compute the digest, any modification of the graphs will result in a different digest. If the search of the block graphs is successful and the computed digest is found, then the log event must have remained unmodified [Anderson, 2016]. Therefore, the auditor must recompute the integrity proof digest of the log event (using the same method as the logger) as described in Sect. 3.2. A SPARQL query is then executed against the block graphs to find a matching digest. If the query returns a block graph containing the integrity proof digest, we proceed to verify the signature in the signature graph. If no matching value is found in the block graphs, we conclude that the integrity of the log event has been compromised. In the case of no matching integrity proof digest or signature verification failure, the auditor will know which event has been modified and who the logger of the event is. However, the auditor will not know *what* the modification is, only that a modification *has occurred*. Therefore, proof of malicious interference would need further investigation.

Despite the verification process supporting the confidentiality, authenticity and integrity of a privacy log, the approach is susceptible to an *internal* attack to subvert the verification process. However, to be successful, an attacker would have to generate and sign a *fake* log event, store the event in the quad store, calculate an integrity proof, store the proof on the blockchain, and finally generate a block graph pointing to the fake integrity proof block.

4 Experimental Evaluation

This section presents a scalability evaluation of our blockchain enabled privacy audit log model from an auditor's perspective. In the experiment, we ran our log integrity verification procedure on increasingly sized L2TAP audit logs.

To simulate the process of an auditor checking the integrity of an audit log, we generated synthetic L2TAP logs[1], which consist of events such as, log initialization, privacy preferences and policies, access request, access response, and actual access; specific event content can be found in [Samavi and Consens, 2018]. The signature and block graph for each event needs to be generated for the auditor to perform the integrity verification procedure. The initial state of the experiment is an audit log containing n events (composed of $2n$ header and body graphs) with n generated signature and block graphs. All of these graphs ($4n$) are stored in a quad store prior to measuring the scalability of the integrity verification solution. Figure 4 depicts the log sizes in the experiment, which included 10,000 events with 100,000 triples.

The experiment was run by executing SPARQL queries on a Virtuoso[2] server and quad store deployed on a Red Hat Enterprise Linux Server release 7.3 (Maipo) with two CPUs (both 2 GHz Intel Xeon) and 8 GB of memory. The RDF graph processing and hash computations for integrity proof and signature computations were run on a MacBook Pro with a 2.9 GHz Intel Core i7 processor and 8 GB of memory. The execution time measures the time difference between sending the queries to the quad store on the server over HTTP and verifying the integrity proof digest and the signature. The recorded time does not take into account the time to generate the signature and block graphs (these were pre-computed before the experiment) or the time needed to write the data to the Bitcoin blockchain. To account for variability in the testing environment, each reported elapsed time is the average of five independent executions.

The elapsed execution time is plotted in Fig. 4. The graph illustrates the execution time of verifying the signature, computing and verifying the integrity of the events, and the overall process. The experiment validates the linear time growth for the entire integrity checking procedure. In practice, an auditor would operate on a subset of events in the log based on the results from compliance queries for a given access request. We have opted to demonstrate a `worst-case` scenario by verifying the integrity and authenticity of an *entire* log rather than a subset of the events. This also demonstrates the execution time of large subsets of events that are the size of the entire log.

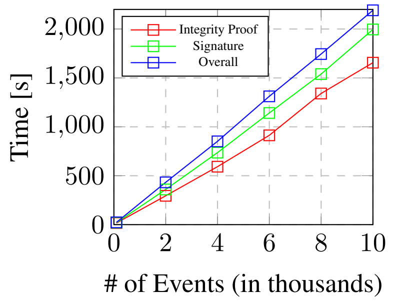


Figure 4: Execution times for integrity and signature verification

5 Related Work

There are a number of proposals that provide a mechanism for verifying the integrity of an audit log [Accorsi, 2009; Stathopoulos *et al.*, 2008]. Butin *et al.* [Butin *et al.*, 2013] address the issues of log design for accountability, such as how meaningful a posteriori compliance analysis can be performed on the logs. Tong *et al.* [Tong *et al.*, 2014] provide role-based access control auditability in audit logs to prevent the misuse of data. However, these solutions only address the integrity of audit logs and miss the non-repudiation aspect.

Kleedorfer *et al.* [Kleedorfer *et al.*, 2016] propose a Linked Data based messaging system that verifies conversations using digital signatures. The RDF graph messages are signed and a signature graph is produced, which can be iteratively signed as the messages pass between recipients. Kasten *et al.* [Kasten *et al.*, 2014] provide a framework for computing RDF graph signatures. This framework supports signing graph data at different levels of granularity and multiple graph signatures [Kasten *et al.*, 2014]. Kasten [Kasten, 2016] discusses how the confidentiality, integrity, and availability of Semantic Web data can be achieved through approaches of Semantic Web encryption and signatures.

Use of blockchain technology in the auditing of financial transactions have been investigated [Anderson, 2016] after the Enron Scandal, where auditor fraud was the source of public distrust [Spoke, 2015]. Anderson [Anderson, 2016] proposes a method of verifying the integrity of files using a blockchain. Similar to our approach, Cucurull *et al.* [Cucurull and Puiggalí, 2016] present a method for enhancing the security of logs by utilizing the Bitcoin blockchain. Our approach differs by providing a model to create tamper-proof logs in a highly scalable Linked Data environment.

6 Conclusions

In this paper we presented a method for utilizing blockchain technology to provide tamper-proof privacy audit logs. The provided solution applies to Linked Data based privacy audit logs, in which lacked a mechanism to preserve log integrity. The model can be used by loggers to generate tamper-proof privacy audit logs whereas the integrity verification queries can be used by external auditors to check if the logs have

[1]Logs available at: https://doi.org/10.6084/m9.figshare.5234770
[2]https://virtuoso.openlinksw.com

been modified. The paper includes an experimental evaluation that demonstrates the scalability of the audit log integrity verification procedure.

There are a number of directions for future work. First, we acknowledge Bitcoin's limitations in terms of cost, speed, and scalability [Manu, 2017]. We utilized Bitcoin since it provides an established mechanism suitable for integrity proofs and to demonstrate the feasibility of our solution applied to Linked Data. For an optimized implementation, other blockchain technologies should be compared in terms of transaction fee, scalability, and smart contract and private ledger support. Second, a large log will require many transactions and occupy a large space on the blockchain. Merkle trees [Merkle, 1980] can reduce the storage and transaction requirements by writing the tree root (composed of multiple integrity proofs) to the blockchain. However, this increases the effort to verify the log integrity since more hash value computations are required to reconstruct the hash tree. Formalizing the trade-offs between hash trees and the verification effort is an interesting optimization problem to investigate.

Acknowledgments

Support from NSERC and SOSCIP are acknowledged. This paper is an abridged version of a paper titled "Blockchain Enabled Privacy Audit Logs" that was nominated for best research paper and best student research paper awards at the "ISWC 2017" conference.

References

[Accorsi, 2009] Rafael Accorsi. Log data as digital evidence: What secure logging protocols have to offer? In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, volume 2, pages 398–403. IEEE, 2009.

[Anderson, 2016] Nicolai Anderson. Blockchain technology: A game-changer in accounting? *Deloitte, March*, 2016.

[Butin *et al.*, 2013] Denis Butin, Marcos Chicote, and Daniel Le Métayer. Log design for accountability. *IEEE Security and Privacy Workshops*, 2013.

[Carroll *et al.*, 2005] Jeremy J Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):247–267, 2005.

[Cucurull and Puiggalí, 2016] Jordi Cucurull and Jordi Puiggalí. Distributed immutabilization of secure logs. In *International Workshop on Security and Trust Management*, pages 122–137. Springer, 2016.

[Heath and Bizer, 2011] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[Kasten *et al.*, 2014] Andreas Kasten, Ansgar Scherp, and Peter Schauß. A framework for iterative signing of graph data on the web. In *European Semantic Web Conference*, pages 146–160. Springer, 2014.

[Kasten, 2016] Andreas Kasten. *Secure semantic web data management: confidentiality, integrity, and compliant availability in open and distributed networks.* PhD thesis, University of Koblenz and Landau, Germany, 2016.

[Kehoe *et al.*, 2015] Lory Kehoe, David Dalton, Cillian Leonwicz, and Thomas Jankovich. Blockchain disrupting the financial services industry? *Deloitte*, 2015.

[Kleedorfer *et al.*, 2016] Florian Kleedorfer, Yana Panchenko, Christina Maria Busch, and Christian Huemer. Verifiability and traceability in a linked data based messaging system. In *Proceedings of the 12th International Conference on Semantic Systems*, pages 97–100. ACM, 2016.

[Manu, 2017] Sporny Manu. Building better blockchains. In *Linked Data in Distributed Ledgers Workshop Keynote*. WWW2017, 2017.

[Merkle, 1980] Ralph C Merkle. Protocols for public key cryptosystems. In *Security and Privacy, 1980 IEEE Symposium on*, pages 122–122. IEEE, 1980.

[Nakamoto, 2008] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[Pilkington, 2016] Marc Pilkington. Blockchain technology: principles and applications. *Research handbook on digital transformations*, page 225, 2016.

[Rodriguez, 2008] Alex Rodriguez. Restful web services: The basics. *IBM developerWorks*, page 33, 2008.

[Samavi and Consens, 2014] Reza Samavi and Mariano P Consens. Publishing l2tap logs to facilitate transparency and accountability. In *LDOW*, 2014.

[Samavi and Consens, 2018] Reza Samavi and Mariano P Consens. Publishing privacy logs to facilitate transparency and accountability. *Journal of Web Semantics*, 2018.

[Sayers and Karp, 2004] Craig Sayers and Alan H Karp. Computing the digest of an rdf graph. *Mobile and Media Systems Laboratory, HP Laboratories, Palo Alto, USA, Tech. Rep. HPL-2003-235*, 1, 2004.

[Spoke, 2015] Matthew Spoke. How blockchain tech will change auditing for good, 2015.

[Stathopoulos *et al.*, 2008] Vassilios Stathopoulos, Panayiotis Kotzanikolaou, and Emmanouil Magkos. Secure log management for privacy assurance in electronic communications. *computers & security*, 27(7-8):298–308, 2008.

[Swarup *et al.*, 2006] Vipin Swarup, Len Seligman, and Arnon Rosenthal. A data sharing agreement framework. In *International Conference on Information Systems Security*, pages 22–36. Springer, 2006.

[Tong *et al.*, 2014] Yue Tong, Jinyuan Sun, Sherman SM Chow, and Pan Li. Cloud-assisted mobile-access of health data with privacy and auditability. *IEEE Journal of biomedical and health Informatics*, 18(2):419–429, 2014.

[Weitzner *et al.*, 2008] Daniel J Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James Hendler, and Gerald Jay Sussman. Information accountability. *Communications of the ACM*, 51(6):82–87, 2008.