A Framework for Safely Publishing Communication Traces

Abhinav Parate and Gerome Miklau
University of Massachusetts, Amherst
Department of Computer Science
140 Governors Drive, Amherst, MA
aparate@cs.umass.edu, miklau@cs.umass.edu

ABSTRACT

A communication trace is a detailed record of the communication between two entities. Communication traces are vital for research in computer networks and study of network protocols in various domains, but their release is severely constrained by privacy and security concerns. In this paper, we propose a framework in which a trace owner can match an anonymizing transformation with the requirements of analysts. The trace owner can release multiple transformed traces, each customized to an analyst's needs, or a single transformation satisfying all requirements. The framework enables formal reasoning about anonymization policies, for example to verify that a given trace has utility for the analyst, or to obtain the most secure anonymization for the desired level of utility. Because communication traces are typically very large, we also provide techniques that allow efficient application of transformations using relational database systems.

Categories and Subject Descriptors: H.2.m [Database Management]:Miscellaneous; K.4.1 [Computers and Society]: Public Policy Issues - Privacy

General Terms: Security, Design, Performance.

1. INTRODUCTION

A *communication trace* is a detailed record of the communication between two entities. Communication traces arise in a variety of settings and include network traces, phone toll records, instant-messaging transcripts, among others. Each record in a communication trace typically identifies a source and a destination, along with descriptive fields such as time stamp, transmitted content, length of transmission, and communication ports.

Communication traces are vital to research into traffic analysis, communication protocols, routing in networks, and security of communication networks. Unfortunately the public release of communication traces remains highly constrained by privacy and security concerns and the lack of available traces is a serious concern for researchers [4, 13].

The safe release of communication traces is a significant challenge. First, communication traces are transactional in nature, with information about entities spread across multiple records, and correlations between records. Conventional k-anonymization [12, 14] and its extensions for transactional data [15] are focused on conceptions of utility that are inappropriate and insufficient for communication traces. The second challenge to protecting communication traces is their massive size. Many proposed anonymization schemes simply cannot scale to such large data sets.

Our approach In this work we propose an approach to communication trace publication emphasizing utility and scalability. We address the problem faced by a *trace owner* who wishes to allow a group of independent *analysts* to safely study a communication trace.

The first component of our approach is a set of simple, formally-defined *transformation operators* that are applied to the trace to remove or obscure information. These transformation operators can be combined to form composite transformations that can be applied to publish output traces, and can be thought of as a safe *view* of the original trace.

Unlike most approaches to trace anonymization (in which the trace owner generates a single anonymized trace) we provide a framework for the trace owner to anonymize a trace for the needs of a particular analysis, releasing multiple traces. The published traces can be more secure because they provide only the needed information, omitting everything else. Our publication framework is illustrated informally in Figure 1. The figure shows an original trace T transformed in four different ways, for use by different analysts. Trace T_1 contains sufficient information for both analysts A and B. Trace T_2 is devised for use exclusively by the analyst C, and trace T_3 is customized for the needs of analyst D. An alternative to publishing both trace T_2 and T_3 is to derive the single trace T_{23} which can support analysts C and D simultaneously.

The second component of our approach is input from the analyst. We assume the requesting trace analyst provides a description of the information needed for analysis. We propose a simple language for *utility constraints* which express the need for certain relationships to hold between the original trace and the published trace.

The third component of our approach is the formal evaluation of privacy and utility. Because both the transformations and the utility requirements of analysts are specified formally, it is possible for the trace owner to analyze trace publication scenarios precisely. In particular, the trace

CIKM'09, November 2–6, 2009, Hong Kong, China.

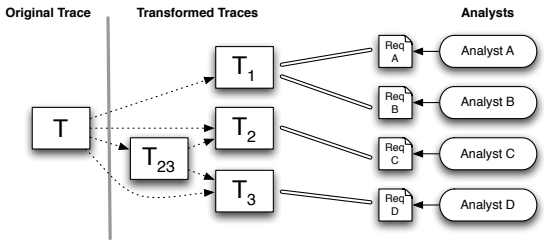

Figure 1: The proposed trace protection framework: the original trace T may be transformed in multiple ways (T_1, T_2, T_3, T_{23}) to support the requirements of different analysts.

owner can: (1) decide whether a composite trace transformation satisfies an analyst's requirements and guarantees perfect utility; (2) compute the most secure transform satisfying a given set of analyst requirements; (3) compare the security and collusion risks of various transforms that can meet requirements of all the analysts.

The result of our contributions is a framework in which basic trace transformation operations can be applied efficiently, and with a precise, formal understanding of their impact on trace utility and privacy. In the remainder of the paper, we describe components of our framework (Section 2), the formal analysis supported by framework (Section 3), the efficient implementation of the system (Section 4) and the related work (Section 5).

2. TRANSFORMATION FRAMEWORK

In this section we describe the two main objects of our framework: *operators*, used by the trace owner to define trace transformations, and *constraints*, used by analysts to express utility requirements.

2.1 Trace transformation operators

The following transformation operators are applied to a trace in order to obscure, remove, or translate field values, making it more difficult for an adversary to attack, but also less useful for analysts. The trace owner may combine individual operators to form composite transformations, balancing utility and security considerations. The output of a composite transformation is released to the analyst.

Operator descriptions

We consider a communication trace as a table consisting of *records*. Each record consists of fixed number of fields.

Projection The projection is the simplest operator which is similar to relational projection operator but it retains the duplicates. It is denoted Π_X for retained attributes in X.

Encryption The encryption operator is denoted $E_{X(Y),\kappa}$ where X is a set of target fields whose values are replaced by ciphertext obtained by applying symmetric encryption function on string formed by concatenating values from X and Y, using κ as the key. The values of Y are not affected.

Canonical Ordering The *canonical ordering* operator replaces the values in fields by synthetic values that respect the ordering of the original values. The ordering operator is denoted $O_{X(Y)}$ where X is the set of target fields to be replaced. The optional set Y is used to form the groups agreeing on values in Y. The replaced values of X respects the ordering only within the group.

Translation The translation operator, denoted $T_{X(Y)}$ translates the values in columns of X by adding or subtracting a parameter whose value is determined using a function taking values in Y as input. If the Y is empty set, all records are translated by some constant c.

Scaling The scaling operator is denoted $S_{X,k}$ and it scales the values in target fields X by multiplying with a constant multiplier k.

It is sometimes convenient to consider the identity transformation, denoted I_X, which does not transform field X, including it in the output without modification.

Composite Transformations

The operators above can be combined to form composite transformations for a trace. We assume in the sequel that composite transformations ϕ are represented in the following normal form:

$$\phi = \Pi_X \circ \phi^1_{X_1} \circ \phi^2_{X_2} \circ ... \circ \phi^n_{X_n} \quad (1)$$

where $\phi^i_{X_i}$ refers to $(i+1)^{th}$ operator in ϕ which acts on attribute set X_i and for all i, $\phi^i_{X_i} \in \{E, T, O, S, I\}$. We denote the set of all such transformations Φ. The last operation applied to the trace is the projection Π_X. Any operator acting on fields not present in X will be disregarded. Further we restrict our attention to composite operations in which each field in the trace is affected only by one operation: $\forall i, j, X_i \cap X_j = \{\}$.

Other operators.

We have found that the above simple set of operators can be used to generate safe transformations supporting a wide range of analyses. However, addition of additional operators to our framework is easy and requires only minor extensions to support other features of the framework.

2.2 Specifying Utility Requirements

In our framework, the analyst seeking access to a trace must specify their utility requirements formally. These requirements are expressed as a set of *constraints* asserting a given relationship between fields in the original trace and fields in the anonymized trace. The syntax of notation for the constraint is as follows:

$$\langle qualifier \rangle \Rightarrow (expr(orig) = expr(anon))$$

where $'expr'$ can be any acceptable arithmetic expression obtained using operators $(+, -, /, *)$ or it can be any boolean expression obtained using operators $(\&\&, ||, \leq, \geq, ==, !=)$ on fields in the trace.

The above constraint means that if there are one or more records in a trace that satisfy the boolean qualifying conditions given in $\langle qualifier \rangle$, then the value of expression *expr* evaluated over these records must be equal to the value of same expression when evaluated over corresponding anonymized records.

We believe trace analysts will be able to use these constraint rules to accurately describe the properties of a trace required for accurate analysis. The analyst could be assisted in this task by a GUI or semi-automated procedures, but this is beyond the scope of the current work.

As an example, in Table 1, we semi-formally describe the requirements as a set of constraints for a real study[6] from

Table 1: *A semi-formal description of utility requirements sufficient to support the example TCP/IP network analysis.*

Semi-Formal Utility Requirements	
Any tuple t in trace	Any tuples $t1, t2$ belonging to same connection in trace
PRESERVE($t.syn$)	PRESERVE($t1.seq\_no \leq t2.seq\_no$)
PRESERVE($t.ack$)	PRESERVE($t1.seq\_no - t2.seq\_no$)
PRESERVE($t.window$)	PRESERVE($t1.ts \leq t2.ts$)
	PRESERVE($t1.ts - t2.ts$)
	PRESERVE($t1.seq\_no == t2.ack\_no$)
where PRESERVE($expr$) $\equiv expr_T = expr_{\phi(T)}$ i.e. value of $expr$ evaluated over tuples in trace T must be equal to value when $expr$ is evaluated over transformed tuples in $\phi(T)$	

the area of network research. This study focuses on estimating the round trip time of a network packet using IP-level packet network traces (see Table 2(a) for illustration). An anonymization scheme to support this study can be expressed as a composite transformation function ϕ given by:

$$\phi = \Pi_X \circ E_{\{ip1,ip2\},\kappa_1} \circ E_{\{pt1,pt2\}(ip1,ip2),\kappa_2} \circ T_{\{ts\}(C)} \circ T_{\{seq\_no,ack\_no\}(C)} \circ I_{\{dir,window,syn,ack\}}$$

Here $C = \{ip1, ip2, pt1, pt2\}$. The records in the sample trace given in Table 2(a) are transformed using the above transformation function ϕ, to obtain the anonymized view given in Table 2(b). The encrypted values have been replaced by variables for clarity.

3. ANALYSIS OF TRANSFORMATIONS

In this section, we briefly describe the important feature of the framework which allows the trace owner to reason formally about the anonymizing transformations and utility. In addition, it allows formal comparison of transformations and has its implications in computing *most secure transformation*, comparing alternative publication strategies and analyzing the impact of collusion.

3.1 Utility verification of a transformation

In our framework, it is possible to test efficiently whether a given transformation ϕ will always satisfy the utility requirements expressed by a set of constraints C. Checking utility constraint satisfaction is performed independently for each constraint rule in C by matching the conditions specified in a constraint to the operators that impact the named fields. Recall that the constraint has an expression $expr$ where it can be conjunctive normal form of one or more sub-expressions, or an arithmetic expression. In our framework, we maintain a look-up table consisting of possible sub-expressions along with the satisfying transformations and conditions. For each sub-expression in a constraint, we look for the corresponding entry in look-up table. If the composite transform function ϕ has a matching transformation in the table for each of the sub-expression, then the constraint is said to be satisfied by the transformation. The details of this process and can be found in our technical report [11].

3.2 A partial order for transformations

Since each transformation operator removes information from the trace, some composite transformations can be compared with one another in terms of the amount of information they preserve. It can be shown that there is a natural partial order (denoted $\preceq$) on transformations. Given two transformations ϕ_1 and ϕ_2, we say that ϕ_1 is more *strict* than ϕ_2 or $\phi_1 \prec \phi_2$ if the information preserved by ϕ_1 is contained within the information preserved by ϕ_2.

The precise definition of *strictness* relation and the relations for basic operators are given in our report [11].

Recall that Φ denotes the set of all composite transformations. Then the following theorem show that the strictness relation has a number of convenient properties.

THEOREM 1. *$(\Phi, \preceq)$ is a partially ordered set and forms a join-semilattice i.e. for any two transformations ϕ_1 and ϕ_2, there is another transformation in Φ, denoted lub(ϕ_1, ϕ_2), which is the least upper bound of ϕ_1 and ϕ_2.*

Theorem 1 can be easily extended to conclude that any set of transforms has a unique least upper bound and this fact has a number of important consequences for the trace publisher:

- First, given a set of constraints C it is important for the trace publisher to compute the most secure transformation satisfying C. Theorem 1 shows that such a transformation always exists. The algorithm for computing this transformation is given in our report [11].
- Next, imagine that the trace publisher has derived three transforms ϕ_1, ϕ_2, ϕ_3 specific to three analyst requests. The publisher may wish to consider publishing a single trace that can satisfy all three requests simultaneously. The least upper bound of these three transformations, denoted $lub(\phi_1, \phi_2, \phi_3)$ is the transformation with least information sufficient for all three analysts.
- Similarly, if the publisher has already released the traces derived from ϕ_1, ϕ_2, ϕ_3 and fears that the analysts may collude, then the least upper bound transformation $lub(\phi_1, \phi_2, \phi_3)$ is a conservative bound on the amount of information the colluding parties could recover by working together. The details of accurate quantitative analysis of collusion are given in our report [11].

4. SYSTEM IMPLEMENTATION

Our system allows the trace owner to efficiently transform large traces in response to multiple requests from analysts. We use a relational database to store the original trace and to apply transformations, creating new traces to be released to users.

In this system, the transformation operators from (Π, E, T, S, I), can be applied in a single scan of the trace using SQL queries with user-defined functions (UDFs). The operators E, T, S, can be implemented efficiently using UDFs, and add only modest CPU overhead. The operators I and Π are implemented using costless projection.

On the other hand, the canonical ordering operator $O_{X(Y)}$ can be applied using the DENSE_RANK() function which was added to the SQL:2003 standard. This function computes the rank of the tuples in a relation based on the rank criteria provided in assisting clause. The operation $O_{X(Y)}$ can be done using following clause: DENSE_RANK OVER (PARTITION BY Y ORDER BY X). Unlike scalar UDFs, this function requires sorting of the base relation. Thus, the cost of transformation increases linearly with the number of O-operators due to multiple scans of the trace.

Table 2: (a) Example of IP-level trace (b) Trace transformed under ϕ as described in Section 2

(a)

ts	*ver*	*ip1*	*ip2*	*pt1*	*pt2*	*dir*	*seq_no*	*ack_no*	*window*	*syn*	*ack*
30	4	172.31.1.34	172.31.2.212	22	22	→	5000	7280	8760	0	1
31	4	172.31.1.34	172.31.2.212	22	22	→	5012	7280	8760	0	1
32	4	172.31.1.34	172.31.2.212	22	22	←	7280	5024	65110	0	1
32	4	172.31.1.34	172.31.2.212	22	22	→	5024	7280	8760	0	1
31	4	172.31.1.34	172.31.2.212	80	9080	→	4780	8214	6432	0	1
30	4	172.31.1.34	172.31.2.89	80	9080	→	1000	1280	17424	0	1
31	4	172.31.1.34	172.31.2.89	80	9080	→	1012	1280	17424	0	1
32	4	172.31.1.34	172.31.2.89	80	9080	→	1024	1280	17424	0	1

(b)

ts	*ip1*	*ip2*	*pt1*	*pt2*	*dir*	*seq_no*	*ack_no*	*window*	*syn*	*ack*
0	c_1		p_1		→	0	2280	8760	0	1
1	c_1		p_1		→	12	2280	8760	0	1
2	c_1		p_1		←	2280	24	65110	0	1
2	c_1		p_1		→	24	2280	8760	0	1
0	c_1		p'_1		→	0	3434	6432	0	1
0	c_2		p_2		→	0	280	17424	0	1
1	c_2		p_2		→	12	280	17424	0	1
2	c_2		p_2		→	24	280	17424	0	1

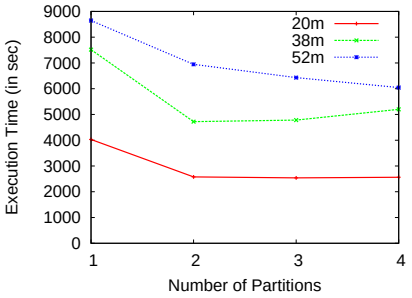

Figure 2: The execution time vs number of partitions graph for varying data sizes varying from 20 million rows to 52 million rows. The query consisted of six rank operators.

In order to optimize the transformation cost for canonical ordering, we use vertical partitions of the trace where each partition has fewer columns and is customized for specific O operators. The vertical partitions prevents redundant sorts of the various columns in trace and have low sorting costs due to smaller size. The results from different partitions are merged to provide a transformed trace. Figure 2 shows the reduction in execution time of a query with 6 dense rank functions as the number of vertical partitions increases.

Further optimization can be done by storing the ranks computed for a query in auxiliary tables and using them later for another query. The details of cost estimation, designing partitions for the workload and using auxiliary tables can be found in our technical report [11].

5. RELATED WORK

K-anonymity [12, 14] and variants [7] apply to the case where there is exactly one record per entity in the data. Terrovitis et al., have extended the definition of k-anonymity for the privacy-preserving publication of set-valued data containing multiple entries for the same entity [15]. Verykios et al [16] considered the privacy of transactional data in the context of data-mining where they wanted to prevent subsets of association rules from being learned. The communication traces are set-valued and transactional in nature. However, communication trace anonymization is significantly different because conventional analysis cannot depend on subtle ordering and correlation among entries.

For the special case of network traces, anonymization has received special attention by researchers, with IP packet traces the most common case. Proposed anonymization techniques include *tcpurify* [3], the *tcpdpriv* [2] and Crypto-PAn [5] tools (which can preserve prefix relationships among anonymized IP addresses), as well as frameworks for defining transformations, such as *tcpmkpub* [10]. The focus of these works is on IP address anonymization and do not analyze the utility of the transformations.

Slagell et al [13] recognized the need for a framework to support the trade-off between the security and utility of traces and provide multiple levels of anonymization. The framework proposed in [8] tries to achieve such a balance between utility and privacy but it is restricted to secure queries with aggregations. In [9], the authors require an analyst to write the analysis program in the language supported by framework, but it has to be reviewed by the experts for privacy issues. The PREDICT [1] repository has been established to make network traces available for research. The access to respository is authorized only after the purpose and identity of researchers is reviewed and verified manually.

Acknowledgements: This work was supported by NSF CNS 0627642, NSF DUE 0830876 and NSF CAREER Award 0643681.

6. REFERENCES

[1] Predict. *https://www.predict.org/.*
[2] Tcpdpriv. *http://ita.eee.lbl.gov/html/contrib/tcpdpriv.html.*
[3] Tcpurify. *http://irg.cs.ohiou.edu/ eblanton/tcpurify.*
[4] K. Claffy. Ten things lawyers should know about internet research. *http://www.caida.org/publications/papers/2008/lawyers_top_ten/.*
[5] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon. Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Comput. Netw.*, 46(2):253–272, 2004.
[6] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring tcp connection characteristics through passive measurements. In *Proceedings of INFOCOMM*, 2004.
[7] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.
[8] J. Mirkovic. Privacy-safe network trace sharing via secure queries. In *NDA '08*, pages 3–10. ACM, 2008.
[9] J. C. Mogul and M. Arlitt. Sc2d: an alternative to trace anonymization. In *MineNet '06*, pages 323–328, 2006.
[10] R. Pang, M. Allman, V. Paxon, and J. Lee. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, 36(1):29–38, January 2006.
[11] A. Parate and G. Miklau. A framework for safely publishing communication traces. *Umass Computer Science Technical Report 2009-040*, 2009.
[12] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, 2001.
[13] A. Slagell and W. Yurcik. Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization. In *SECOVAL*, pages 80–89, 2005.
[14] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness and KB Syst.*, 10(5):557–570, 2002.
[15] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *VLDB*, 1(1):115–125, 2008.
[16] V. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering*, 16:434–447, 2003.